E.U.I.T. TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA
PLAN 2000

TEMA: Herramienta desarrollada en Matlab para la autoevaluación de prácticas

TÍTULO: Desarrollo de herramientas para la autoevaluación del Laboratorio de Procesado Digital de la Señal

AUTOR: Mónica Morán Gómez

TUTOR: David Osés del Campo **Vº Bº.**

DEPARTAMENTO: ICS

Miembros del Tribunal Calificador:

PRESIDENTE: Alfonso Martín Marcos

VOCAL: Luis Arriero Encinas

VOCAL SECRETARIO:

DIRECTOR:

Fecha de lectura:

Calificación: **El Secretario,**

RESUMEN DEL PROYECTO:

El proyecto pretende desarrollar, a través del programa Matlab, una herramienta de autoevaluación de los contenidos más importantes que se abordan en las prácticas de la asignatura Procesado Digital de la Señal. Se trata de que el alumno pueda comprobar que va adquiriendo las destrezas necesarias para superar los exámenes correspondientes al laboratorio.
Para la autoevaluación se pedirá al alumno los resultados obtenidos mediante la resolución del enunciado de las prácticas y posteriormente se compararán estos resultados con los resultados correctos, indicando en cada caso si la respuesta es correcta o no lo es.

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA UNIVERSITARIA DE INGENIERÍA TÉCNICA DE

TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

Desarrollo de herramientas para la autoevaluación del Laboratorio de Procesado Digital de la Señal

Autora:

MÓNICA MORÁN GÓMEZ

Tutor:

DAVID OSÉS DEL CAMPO

Madrid, Junio de 2014

Agradecimientos

A mis padres, por dármelo todo

A mis hermanas, por las manos tendidas

A David, mi tutor, por su tiempo y dedicación

A Mercedes, por tener siempre las palabras adecuadas

A Pablo y Sandra, por estar a mi lado

A Susana, por ser incondicional

A Fer, por no dudar

A Nuria y Ximena, por confiar con los ojos cerrados

A Óscar, por el apoyo

A los Cochinos, por entender mis ausencias y mis malos días

A mis compañeros del 2M y electrónicos, que se convirtieron en amigos, por las horas de clase, de biblioteca, de laboratorio, de risas y momentos duros

A los VIP, por convertirse en parte importante de mi vida

A Elena, por entenderme en cada momento y creer en mí

A Ana, por toda la ayuda y los grandes momentos

A Dunnia y David por tirar de mí

A ti, que estás dedicando tu tiempo a leer mi trabajo

No hay palabras suficientes para expresar todo mi agradecimiento:
¡GRACIAS!

“Todos somos unos genios,
pero si juzgas a un pez por su habilidad
para escalar un árbol,
vivirá su vida entera creyendo que es un estúpido”

Albert Einstein

Resumen

Este proyecto tiene como objetivo el desarrollo de una herramienta que permita al alumno la autocorrección de prácticas de la asignatura de Procesado Digital de la Señal.

La herramienta será diseñada por medio del GUI de Matlab, que permite la creación de interfaces gráficos de usuario para la interacción con el alumno, así él mismo podrá comprobar si los resultado obtenidos para el enunciado de la práctica facilitado son correctos.

La evaluación del alumno se llevará a cabo pidiendo distintas respuestas sobre las prácticas y comparándolas posteriormente con los resultados correctos. El código invisible al usuario será el encargado de indicar si el resultado es correcto o no lo es.

Abstrac

The aim of this project is to develop a tool for the students of Digital Signal Processing that helps them self-correct their lab exercises.

The tool will be designed using the Matlab GUI, which allows the creation of graphical user interfaces to interact with the student, who can check whether the results obtained are correct or not.

The student will be asked about different results of the exercises and the answers will be compared with the correct results. A part of the tool hidden to the student will reveal to the lecturer the outcome of this comparison.

Índice

Capítulo I:
El método e-learning

1. Introducción

No existe una única definición para el método e-learning, sin embargo se considera como oficial la dada por la Comisión Europea:

"La utilización de las nuevas tecnologías multimedia e internet, para mejorar la calidad del aprendizaje facilitando el acceso a recursos y servicios, así como los intercambios y la colaboración a distancia"

El método e-learning es un tipo de enseñanza que dista de los métodos tradicionales de enseñanza, ya que con la llegada de internet los métodos de enseñanza-aprendizaje han evolucionado acompañando a los nuevos tiempos. La diferencia más importante con el método de enseñanza tradicional se encocuentra en que el tiempo y el espacio ya no supone un problema, el alumno es quien decide donde y cuando. Debido a esta flexibilidad, el alumno que utiliza esta herramienta puede acceder a ella independientemente de la hora o el lugar desde el que lo haga, lo único que debe tener es un ordenador y una conexión a internet, ya sea pública o privada.

Las vías para el desarrollo del método e-learning son internet o las intranets, por este motivo, también se puede definir como educación a distancia o teleenseñanza. Debido a la constante actualización de internet y su fácil acceso por parte de profesores y alumnos los contenidos se renuevan constantemente, por lo que ningún alumno quedará desactualizado.

Como conclusión de esta breve introducción al método e-learning se puede decir que este método es un producto de las nuevas formas de comunicación situadas en la era digital en la que nos encontramos y se puede afirmar que este método se está situando en un lugar importante entre los actuales métodos de enseñanza/aprendizaje por lo que ha sido mencionado anteriormente, el tiempo y el espacio se convierten en un obstáculo salvado por este método.

2. Historia: De la educación a distancia a la teleenseñanza

Sobre el inicio del método e-learning existen varias teorías; muchos dirían que nació con la llegada del ordenador, otros que con la invención de internet, es necesario remontar mucho tiempo atrás y fijar la vista en los cursos por correspondencia. Estos cursos fueron creados para que la educación llegase a los lugares más lejanos, en las cuales no podían construirse colegios, así los alumnos podían ser supervisados por sus propios padres en esos cursos por correspondencia.

Navegando por internet encontramos que la educación a distancia comenzó en México en 1945, cuando se creó el Instituto Federal de Capacitación del

Magisterio, un año después se creó la Universidad de Sudáfrica. En Gran Bretaña, en 1969, se crea la Open University, y 4 años después en España asistimos a la creación de la Universidad Nacional de Educación a Distancia(UNED), y finalmente, en 1974 se funda en Alemania la FernUniversität Hagen.

De este modo se creó la base del actual método e-learning, que se apoya en estos cursos por correspondencia pero que dista en algunos aspectos de éstos. Las claves para diferenciar uno y otro se encuentran en que el método e-learning ofrece una modalidad semipresencial.

Se dice que un curso es semipresencial cuando profesor y alumno coinciden en el mismo espacio-tiempo, es decir, interaccionan en algunas ocasiones, pudiendo ser mediante chats, foros o la más común: las tutorías. El alumno que se decide por el método e-learning, cuenta además, con las posibilidad de tener una evaluación continúa basada en ejercicios o prácticas que se reciben y envían a través de la web, por el contrario, la educación a distancia o por correspondencia cuenta con un único examen global.

3. Diferencias con la educación presencial tradicional

En el libro publicado por Julio Cabero y Eloy López en 2009 y titulado "*Evaluación de materiales multimedia en red en el espacio europeo de educación superior (EEES)*" se muestran las principales diferencias entre la formación presencial tradicional y la formación basada en la red. En la tabla adjunta, sacada de este libro se pueden observar algunas de sus diferencias:

Tabla 1. Diferencias entre formación presencial y formación basada en la red

FORMACIÓN BASADA EN LA RED	FORMACIÓN PRESENCIAL TRADICIONAL
Permite a los estudiantes que vayan a su propio ritmo de aprendizaje	Parte de una base de conocimientos y el estudiante debe ajustarse a ellas.
Es una formación basada en el concepto de "formación en el momento en que se necesita" (Formación justo a tiempo "Just-in-time training", formación cuando se necesita, donde se necesita y al ritmo marcado por el estudiante)	Los profesores determinan el cuándo y cómo los estudiantes recibirán los materiales formativos.
Permite la combinación de diferentes materiales (impresos, auditivos, audiovisuales) para alcanzar una enseñanza multimedia.	Parte de la base de que el sujeto recibe pasivamente el conocimiento para generar actitudes innovadoras, críticas e investigadoras.

Con una sola aplicación se puede atender a un mayor número de estudiantes.	Suele tender a apoyarse en materiales impresos y en el profesor como fuente de presentación y estructuración de la información.
Su utilización tiende a reducir el tiempo de formación en las personas.	La comunicación se desarrolla básicamente entre el profesor y el estudiante.
Tiende a ser interactiva, tanto entre los participantes en el proceso (profesor y estudiantes) como los contenidos.	La enseñanza se desarrolla de forma preferentemente grupal.
La formación tiende a realizarse de forma individual, sin que ello signifique la renuncia a la realización de propuestas colaborativas.	Puede prepararse para desarrollarse en un tiempo y en un lugar.
Puede ser utilizada en el lugar de trabajo y en el tiempo disponible por parte del estudiante.	Se desarrolla en un tiempo fijo y en aulas específicas.
Es flexible.	Tiende a rigidez temporal.

Fuente: Evaluación de materiales multimedia en red en el espacio europeo de educación superior(EEES). (Cabrero y López, 2009)

3.1. El método e-learning y su aportación a la enseñanza.

La enseñanza tradicional se ha visto mejorada con la llegada del método e-learning, lo que se podría llamar la aportación del método a la enseñanza. Por ello, se distinguen una serie de características, las cuales han sido obtenidas del libro *"Aportaciones del e-learning a la innovación educativa."*, escrito en el año 2009 por Juan Pablo Pons, que indican en qué medida la educación tradicional se ha visto mejorada:

- Salva la rigidez espacio-temporal por la cual, algunos estudiantes no podían acceder a la parte presencial de la educación.
- Ayuda al estudiante en su proceso de autosuficiencia, el proceso de aprendizaje es, en gran medida, responsabilidad suya.
- Elimina las distancias entre profesor-alumno.
- Es el alumno quien decide donde y cuando se lleva a cabo su formación.
- Se dispone de un material didáctico más amplio siempre a disposición del alumno.

El e-learning seguirá evolucionando en los próximos años, debido a la constante evolución de las telecomunicaciones. Esta enseñanza resuelve los

problemas a los que la enseñanza tradicional no es capaz de dar solución.

3.2 Modalidades del método e-learning

El método e-learning no es sólo el que se basa en una educación puramente a distancia y a través de un ordenador. La educación a distancia ha ido evolucionando y adaptándose a los nuevos tiempos y a las nuevas necesidades del entorno en el que no encontramos.

Por lo tanto, a continuación se van a describir algunos de los métodos que han evolucionado desde el método e-learning:

- Método B-learning: este tipo de método es un tipo de aprendizaje mixto, es decir, semipresencial. Combina las clases presenciales con el aprendizaje puramente e-learning.
- Método M-learning: método de aprendizaje de tipo móvil, se basa en dispositivos móviles con capacidad para enviar y recibir mensajes o conectados a internet. Con la llegada de los smartphones y tablets este tipo de educación ha visto como sus usuarios han aumentado. Requiere de una menor inversión por parte del alumno (precio inferior al de un ordenador personal)

Existen tantas vertientes del método e-learning como alumnos con distintas necesidades en relación a su situación personal. Este tipo de método se adapta al estudiante facilitando su aprendizaje.

4. El método e-learning en la EUITT

En la actualidad nuestra Escuela cuenta con una plataforma de enseñanza virtual; esta plataforma lleva por nombre Moodle.

4.1 La plataforma virtual: Moodle

Moodle fue creado por Martin Dougiamas, un australiano graduado en Ciencias de la Computación y la educación, sus siglas responden a:

"Module Object-Oriented Dynamic Learning Environment (*Entorno Modular de Aprendizaje Dinámico Orientado a Objetivos*)"

La idea de Martin Dougiamas era que la educación girara entorno al estudiante y no a los libros de texto, así se consigue que el estudiante desarrolle sus propias habilidades y

no adopte las transmitidas por el profesor a través de un libro de texto como propias

La primera versión de la herramienta vio la luz el 20 de agosto de 2002. Después de esta primera versión, las actualizaciones han ido sucediéndose de forma regular hasta nuestros días.

La plataforma Moodle es usada por profesores y alumnos; los usos más comunes son:

- Los profesores cuelgan en la plataforma virtual transparencias utilizadas en clase, enunciados de prácticas a realizar en la asignatura o ejercicios a realizar por el alumno.
- En algunos casos, el alumnos debe usar esta misma plataforma para entregar estudios previos de prácticas, las memorias de las mismos o ejercicios realizados dentro de la evaluación continúa de la asignatura.
- Realización de test de conocimientos, los profesores tienen la opción de realizar exámenes usando la plataforma virtual, pudiendo regular el tiempo para la realización del test o el número de intentos que cada alumno puede realizar.
- La plataforma Moodle permite el uso de foros, tanto por parte de alumnos como por parte de profesores, pueden ser usados para resolver dudas o para avisar de cambios en horarios de clases o de fechas de entrega prácticas y ejercicios.

Se puede considerar que la plataforma virtual Moodle es de gran ayuda en los modelos de enseñanza actuales, ya que facilita la comunicación profesor-alumno, y permite la entrega bidireccional de material educativo.

Uno de los problemas que puede presentar la plataforma es el error en los servidores, cuando alguno de sus ellos deja de funcionar las funciones de la plataforma quedan inhabilitadas, dejando a profesores y alumnos “desenchufados”.

4.2 El método e-learning en el Laboratorio PDS

Como se ha comentado anteriormente, la Escuela cuenta con una plataforma virtual, Moodle, que es usada por casi la totalidad de los profesores de la Escuela para llegar hasta sus alumnos, facilitarles la documentación usada en las clases teóricas de sus asignaturas así como en los laboratorios, en el caso de tenerlo.

Los profesores de Procesado Digital de la Señal la utilizan como vía de comunicación con los alumnos matriculados en esta asignatura, de modo que, apoyándose en el método e-learning, hacen llegar a los alumnos el enunciado de

las prácticas de forma previa al desarrollo de la sesión. De esta manera, disponen de la documentación necesaria para realizar la misma. Cualquier alumno con ordenador personal y conexión a internet puede obtener el enunciado y administrar su tiempo para la realización de ésta, no siendo necesario acudir al laboratorio para llevarla a cabo.

Con el fin de ayudar al alumno a conocer de forma rápida y sencilla si los resultados obtenidos al resolver la práctica, se han creado las herramientas sobre las que versa este Proyecto Fin de Carrera y que más adelante se explicarán.

5. Ventajas y desventajas del método e-learning

José Antonio Campos, en el año 2003, en el XIV Congreso Internacional de Burgos, habló sobre las principales ventajas y desventajas del método e-learning, en el documento, obtenido de la biblioteca del Centro Virtual Cervantes titulado: "*E-learning e internet como herramientas de autor para profesores de internet*", encontramos los siguientes puntos:

5.1 Ventajas del método e-learning.

- Es posible masificar los procesos de enseñanza y aprendizaje.
- Es posible respetar los ritmos propios de cada alumno, sin retrasar a unos o acelerar a otros.
- No se requieren los niveles de infraestructura de los sistemas formales, con la consiguiente reducción de costos (en tiempos y dinero).
- Se estimula la iniciativa individual y se produce una mejor selección.
- Permite la participación activa de todos los alumnos/as.
- Supervisión continua del tutor.
- Práctica de técnicas de computación, información y comunicación.
- Posibilidad de concentrar las clases directas, tanto en cuanto a contenidos como a tiempo de aprendizaje.
- Posibilidad de supervisar a un gran número de alumnos separados geográficamente.
- Cursos diseñados en base a los alumnos/as.
- Forma de enseñanza orientada hacia el futuro.

5.2 Desventajas del método e-learning.

- Es necesario un equipo informático lo cual supone una inversión económica y de aprendizaje en nuevas habilidades para manejar el sistema de forma autónoma (tanto para el profesor como para el alumno).

- La reducción, o pérdida, de los procesos de socialización en el aula y el grupo.
- La falta de respuesta inmediata a los interrogantes que surgen en el alumno.
- El hecho de que, con frecuencia, se mantienen estilos y pautas de la educación en aula que son disfuncionales para la educación a distancia.
- La relativa escasez de instrumentos didácticos diseñados para, y adecuados a la propuesta.

Capítulo II:
Filtros FIR

1. Introducción

En este capítulo del proyecto, abordaremos el proceso de diseño de filtros digitales, en concreto filtros cuya respuesta al impulso es de una longitud finita (FIR).

Cuando se diseña un filtro FIR es porque queremos explotar una de las características destacables de este tipo de filtros, obtener una respuesta de fase lineal, garantizando así la ausencia de distorsión de fase. Si bien en la mayor parte de las aplicaciones de filtrado usualmente nos preocupa más la discriminación en frecuencia dada por la respuesta en amplitud, existen numerosas aplicaciones en que se requiere un filtro libre de distorsión de fase, y en estos casos sólo cabe el empleo de un filtro FIR de fase lineal. Es por ello que se llevará a cabo un estudio concreto de las características de este tipo de filtros.

El método fundamental que se emplea cuando se desea diseñar un filtro FIR es el método de la ventana o método de la Transformada de Fourier. Este método se basa en el truncamiento de la respuesta al impulso de un sistema ideal de modo que obtengamos finalmente un número finito de muestras aproximándonos, por tanto, a la característica ideal mediante un filtro FIR.

2. Pasos en el diseño de un filtro digital

El diseño de un filtro digital involucra los siguientes pasos:

- Especificación de las características del filtro, es decir, determinar la plantilla de especificaciones que deseamos que cumpla el filtro.
- Elección del método de diseño del filtro y de, en su caso, la aproximación a las especificaciones empleando un sistema de tiempo discreto (filtros IIR)
- Cálculo de los coeficientes del filtro mediante el método seleccionado.
- Representación del filtro mediante una estructura adecuada (realización).
- Análisis de los efectos de longitud de palabra finita sobre el comportamiento del filtro, comprobando que en tales circunstancias el filtro cumple con las especificaciones de partida. En caso contrario habrá que modificar el filtro para que las cumpla.
- Realización prácticas del filtro en software o hardware.

Cuando se emplea un filtro de tiempo discreto para procesado digital de señales continuas, las especificaciones tanto del filtro discreto como el filtro continuo efectivo están dadas típicamente (pero no siempre) en el dominio de la frecuencia. Esto es especialmente común para filtro selectivos en frecuencia como paso bajo, paso banda y paso alto.

3. Propiedades de los filtros FIR

i. Los filtros FIR son una clase de filtros cuyas características quedan determinadas por las siguientes propiedades:

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$H(z) = \sum_{k=0}^{N-1} h[k]z^{-k}$$

$$H\left(e^{j\Omega}\right) = \sum_{k=0}^{N-1} h[k]e^{-j\Omega k}$$

donde h[k], k=0,1,…,N-1, son los coeficientes de la respuesta al impulso de N coeficientes (no nulos), H(z) es la función de sistema (función de transferencia) y $H\left(e^{j\Omega}\right)$ es la respuesta en frecuencia del filtro. La primera es la ecuación en diferencias lineales de coeficientes constantes (EDLCC), que nos describe su comportamiento en el dominio del tiempo. La EDLCC describe el filtro FIR en su forma no recursiva.

ii. Los filtros FIR descritos en la forma anterior (mediante la EDLCC) son siempre estables.

iii. Los filtros FIR pueden tener respuesta de fase exactamente lineal.

iv. Son sencillos de realizar en la práctica. Todos los DSP disponibles tienen arquitecturas apropiadas para realizar filtros FIR. Los filtros FIR no recursivos sufren menos de los efectos de longitud de palabra finita que los IIR. También existen los FIR recursivos, que pueden ofrecer ventajas computacionales significativas.

Los filtros FIR se emplean cuando se desea explotar alguna de las ventajas anteriores, en particular la de fase lineal.

Debido a su método más habitual de realización no recursiva, se les denomina filtros de convolución (obsérvese la expresión de la EDLCC), y desde el punto de vista de la operación que realizan en el dominio del tiempo se les denomina filtros de media móvil o de media móvil ponderada.

La expresión correspondiente a la ecuación en diferencias puede interpretarse como una suma móvil ponderada de N muestras de la señal de entrada x[n], para calcular la secuencia de salida y [n].

4. Filtros FIR de fase lineal.

Cuando una señal pasa por un filtro, es modificada en amplitud y/o fase. La naturaleza y magnitud de la modificación de la señal depende de las características de amplitud y fase del filtro. El retardo de fase o retardo de grupo del filtro proporciona una medida útil de cómo modifica el filtro las características de fase de la señal. Si consideramos una señal compuesta de diversas componentes espectrales, el retardo de fase del filtro es la cantidad de tiempo que es retardada cada componente de frecuencia de la señal al atravesar el filtro. De otro lado el retardo de grupo es el retardo de tiempo medio que sufren las componentes de frecuencia.

Un filtro con una característica de fase no lineal producirá distorsión de fase en la señal. Esto es debido a que cada componente de frecuencia de la señal se retarda una cantidad de tiempo no proporcional a la frecuencia, alterando por tanto la relación armónica.

La distorsión de fase puede evitarse empleando filtros con característica de fase lineal en la banda de interés.

Dependiendo del tipo de simetría de la respuesta al impulso h[n] y de si M es par o impar, nos encontramos con diferentes tipos de filtros de fase lineal generalizada. Por esta razón es útil, en general, definir cuatro tipos de sistemas FIR de fase lineal generalizada.

Tipo	Simetría	Longitud	Orden
I	Par	Impar	Par
II	Par	Par	Impar
III	Impar	Impar	Par
IV	Impar	Par	Impar

5. Método de la trasformada de Fourier o de la ventana

El empleo de filtros FIR se hace casi exclusivamente en sistemas de tiempo discreto. Es por ello que las técnicas de diseño de filtros FIR están basadas en la aproximación directa de la respuesta en frecuencia de un sistema de tiempo discreto. La mayoría de las técnicas de aproximación de la respuesta en amplitud de un sistema FIR asumen restricciones de fase lineal, evitando problemas de factorización espectral.

El método más sencillo es el denominado método de la ventana o de la transformada de Fourier. Es un método muy flexible que puede aplicarse a

cualquier tipo de respuesta en frecuencia deseada. Este método generalmente comienza con una respuesta en frecuencia que puede representarse como:

$$H_d\left(e^{j\Omega}\right) = \sum_{n=-\infty}^{\infty} h_d[n] e^{-j\Omega n}$$

donde $h_d[n]$ es la correspondiente respuesta al impulso, que puede expresarse en función de la respuesta en frecuencia mediante la transformada inversa de Fourier como:

$$h_d[n] = \frac{1}{2\pi} \int_{(2\pi)} H_d\left(e^{j\Omega}\right) e^{j\Omega n} d\Omega$$

Cuando se emplea este método, se presentan dificultades de tipo práctico.

Muchos sistemas idealizados presentan respuestas en frecuencia con discontinuidades en la frontera entre bandas. Como consecuencia, tienen respuestas al impulso que no son causales y son infinitamente largas. La forma más directa de obtener una aproximación FIR causal de tales sistemas es truncar la repuesta al impulso ideal (es decir, quedarnos con un número finito de muestras de la misma).

Nos vamos a encontrar con un compromiso entre tiempo y frecuencia. Si pretendemos tener precisión máxima en frecuencia (corte abrupto entre banda de paso y banda atenuada), la respuesta al impulso debe tener una duración finita, extendiéndose a izquierda y derecha en el eje de tiempos (correspondiente a un sistema no causal).

Como se ha dicho, la forma más obvia de aproximar la respuesta en frecuencia mediante un filtro FIR es truncar la respuesta al impulso. Esto es, quedarnos con los valores que toma la repuesta al impulso en un cierto intervalo de tiempo finito. Si este intervalo incluyera valores correspondientes a instantes negativos de tiempo (que correspondería a un sistema no causal), puede realizarse un desplazamiento en el tiempo de magnitud suficiente para que el filtro aproximado sea causal (el desplazamiento introduciría un término de fase línea). Cuantas más muestras incluyamos en la respuesta al impulso aproximada h[n], mejor se aproximara su respuesta en frecuencia a la partida (la ideal). El problema es que a mayor número de coeficientes el coste aumenta (es necesaria más memoria), la carga computacional aumenta (con lo cual la frecuencia de muestreo debe disminuir, disminuyendo el rango de frecuencia de aplicabilidad), y tendrá un mayor retardo.

5.1 Ventana rectangular.

El empleo de la función ventana rectangular corresponde al caso de truncar directamente la respuesta al impulso deseada $h_d[n]$. La respuesta al impulso aproximada resultante esta dada por

$$h[n] = h_d[n] \cdot w[n]$$

$$w[n] = \begin{cases} 1 & -\frac{M}{2} \leq n \leq \frac{M}{2} \\ 0 & resto \end{cases}$$

Es conveniente trabajar con secuencias simétricas con respecto al origen (como se ha indicado en la expresión de w[n]), para que los espectros de las mismas sean reales, aunque los sistemas no sean causales (o las secuencias empleadas). Posteriormente, para que la respuesta al impulso aproximada h[n] sea causal, se desplaza la misma hacia la derecha, dando lugar a una repuesta en frecuencia que no será real pero tendrá una característica de fase línea (retardo de fase constante).

La transformada de Fourier de la ventana rectangular, definida mediante la expresión anterior, es

$$W(e^{j\Omega}) = \frac{sen[\Omega(M+1)/2]}{sen(\frac{\Omega}{2})}$$

Algunos ejemplos de ventanas rectangulares podrían ser los siguientes:

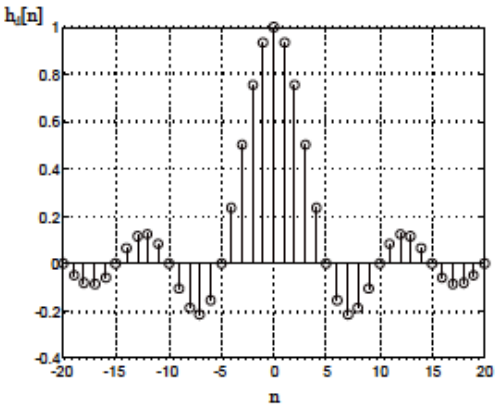


figura 1

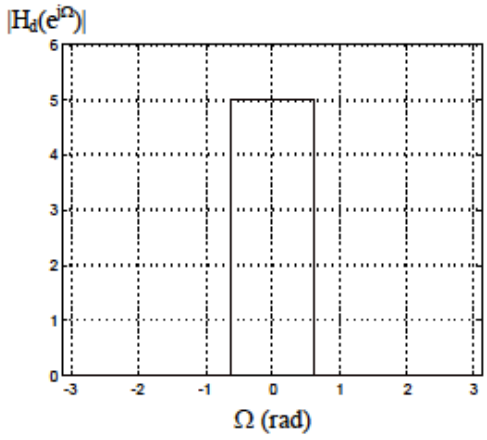


figura 1.1

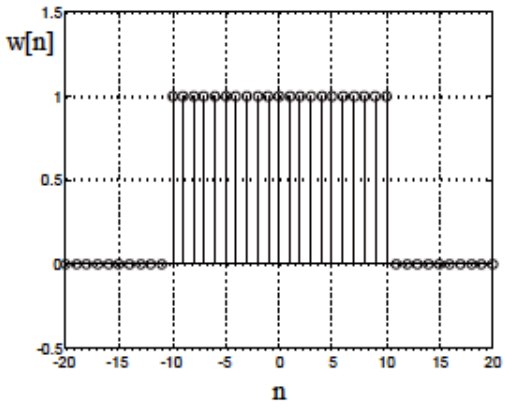


figura 2

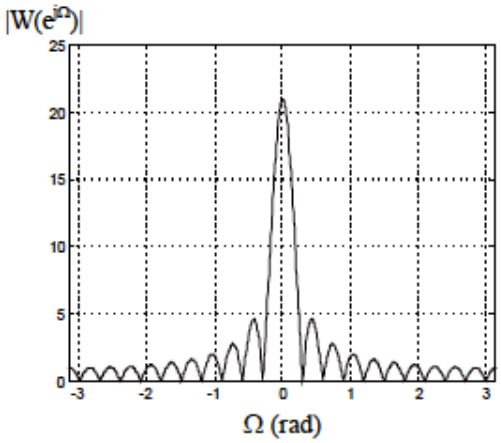


figura 2.2

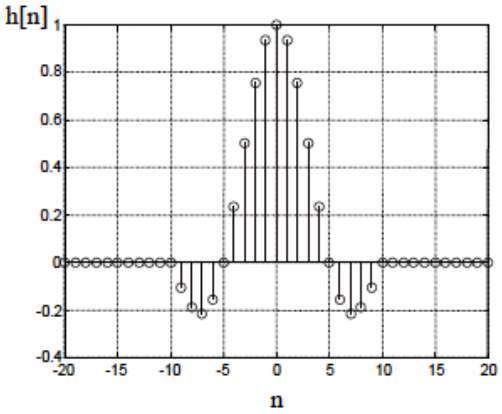


figura 3

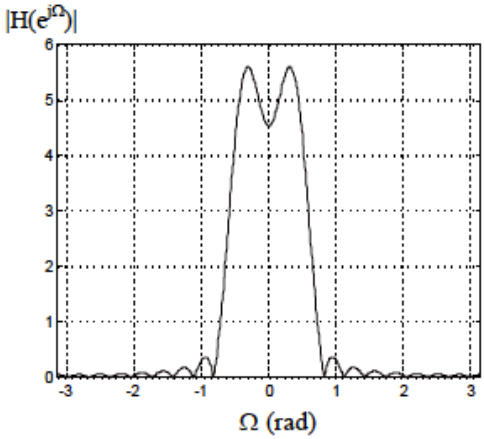


figura 3.3

Para comparar la amplitud de los lóbulos laterales de las distintas ventanas es conveniente utilizar una escala logarítmica en amplitudes, ya que los niveles de estos lóbulos son muy bajos y pueden no apreciarse en una escala lineal. A continuación, podemos observar la comparación entre las transformadas de Fourier de ventanas rectangulares de distintas longitudes:

L=11

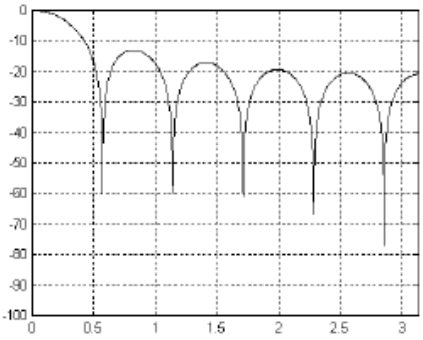


figura 4

L=41

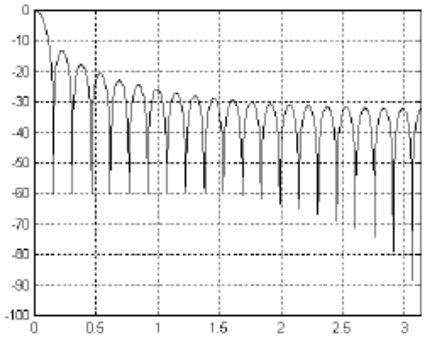


figura 4.4

5.2. Ventana triangular o de Bartlett.

La ventana de Bartlett aumenta la diferencia entre el lóbulo principal y primer lóbulo secundario frente a la ventana triangular (-13,5dB en la ventana rectangular, mientras que en la de Bartlett son -27dB) a consta de ensanchar el lóbulo principal (para una misma longitud de las ventanas)

$$w[n] = \begin{cases} \frac{(M+1)-|n|}{(M+1)^2} & -M \le n \le M \\ 0 & resto \end{cases}$$

Podemos ver esta ventana como la convolución de dos ventanas rectangulares iguales de longitud N = M+1 muestras, con lo que se obtiene una ventana resultante 2M+1 muestras. Según la propiedad de convolución de la transformada de Fourier, es espectro de la ventana de Bartlett está dado por

$$W(e^{j\Omega}) = \left(\frac{sen[\Omega(M+1)/2]}{sen(\frac{\Omega}{2})} \right)^2$$

A continuación se muestra la ventana de Bartlett

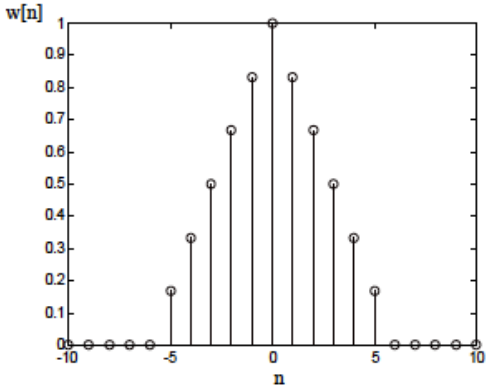


figura 5

5.3. Ventana de Hanning.

Como se ha dicho anteriormente y ha quedado patente en el estudio de las ventanas rectangular y Hanning debe alcanzarse un compromiso entre selectividad del filtro deseado (lóbulo principal de la transformada de Fourier de la función ventana estrecho) y el rizado en las bandas de paso y atenuadas (amplitud de los lóbulos laterales pequeña). Cuando el rizado no es un inconveniente, la ventana rectangular proporciona la máxima selectividad para un orden N dado.

Entre las muchas funciones ventana desarrolladas, las ventanas de Von Hann y de Hamming han recibido una aceptación considerable entre los diseñadores de filtros digitales debido a que presentan una anchura de lóbulo principal similar a la ventana de Bartlett pero con un menor nivel de lóbulos laterales (para una misma longitud de ventanas).

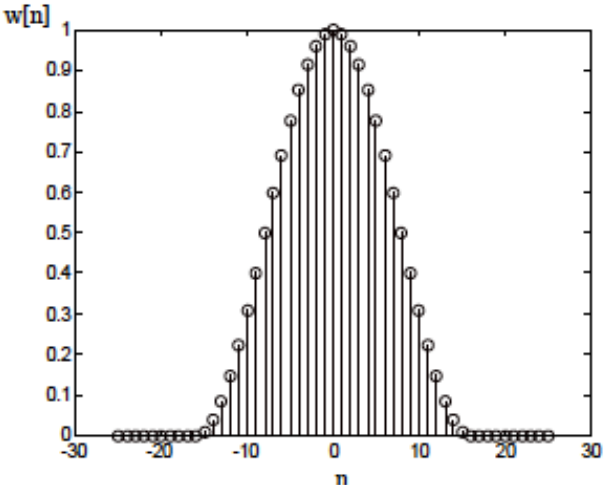


figura 6

La ventana de Hanning de 2M+1 muestras tiene la siguiente expresión:

$$w[n] = \begin{Bmatrix} 0.5 + 0.5cos\left(\frac{n\pi}{M+1}\right) & -M \leq n \leq M \\ 0 & resto \end{Bmatrix}$$

5.4. Ventana de Hamming.

R. W. Hamming descubrió que con una variación en los coeficientes de la ventana de Hann podía reducir drásticamente la amplitud de los lóbulos laterales respecto al principal. La función ventana de Hamming, denominada también coseno alzado con pedestal, se define como

$$w[n] = \begin{Bmatrix} 0.54 + 0.46cos\left(\frac{n\pi}{M}\right) & -M \leq n \leq M \\ 0 & resto \end{Bmatrix}$$

Representación de la función de la ventana de Hamming, que puede compararse con la de Hanning para observar diferencias, así como su transformada de Fourier.

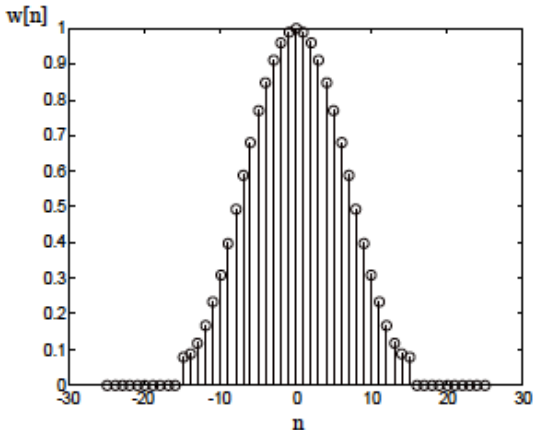


figura 7

En la siguiente ventana podemos observar una comparación entre las diferentes ventanas vistas hasta el momento:

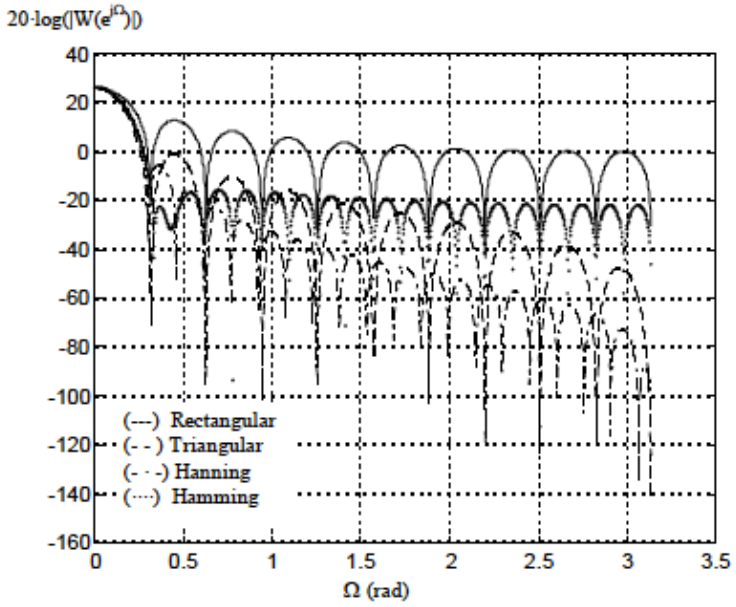


figura 8

5.5. Ventana de Kaiser.

El compromiso entre anchura de lóbulo principal y lóbulos laterales puede cuantificarse buscando la ventana que está más concentrada alrededor de $\Omega = 0$ en el dominio de la frecuencia. Para las ventanas estudiadas, con una longitud dada, este compromiso es fijo.

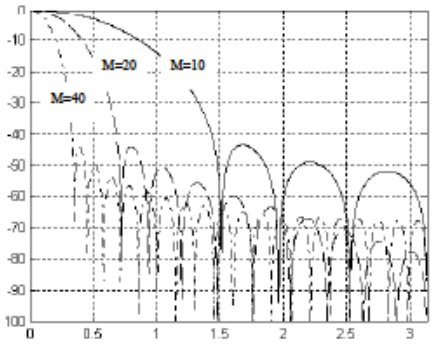
$$w[n] = \begin{Bmatrix} \dfrac{I_o\left[\alpha\left(1-\left(\frac{n}{M}\right)^2\right)^{\frac{1}{2}}\right]}{I_o(\alpha)} & -M \le n \le M \\ 0 & resto \end{Bmatrix}$$

donde $I_{o()}$ representa la función de Bessel modificada de orden cero y primera especie. En contraste con las otras ventanas estudiadas, la de Kaiser tiene dos parámetros: la longitud de la ventana y el parámetro de forma, α. Variando estos parámetros puede ajustarse la amplitud de los lóbulos secundarios para una anchura fija del lóbulo principal. Para n=0, se obtiene w[0]=2 independientemente de los parámetros M y α. Para el caso de $\alpha = 0$, la ventana de Kaiser se reduce a la función de ventana rectángular. Con $\alpha = 5.44$ se obtiene una ventana muy similar a la de Hamming.

El aumento de M, manteniendo α constante, produce un estrechamiento del lóbulo principal, pero no afecta a la amplitud de los lóbulos laterales.
Kaise obtuvo un par de fórmulas que permiten al diseñador predecir los valores de M y α necesarios para cumplir una determinada especificación de un filtro selectivo en frecuencia.

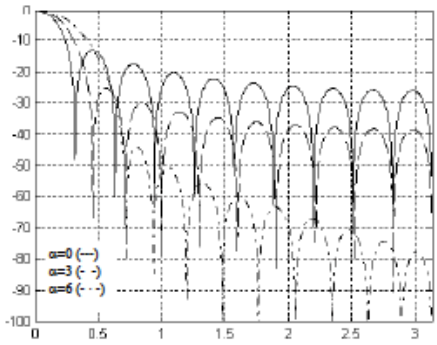
La figura muestra la comparación entre diferentes ventanas de Kaiser:

a) Para distintas longitudes con $\alpha = 6$ y b) para distintos valores de α con L=20



a)

figura 9



b)

figura 9.1

En la tabla 2 mostrada a continuación se muestra la comparación entre las diferentes ventanas. En dicha tabla puede observarse el valor relativo de la amplitud del primer lóbulo secundario respecto del principal, y el error de aproximación, que nos va a dar la máxima atenuación que puede conseguirse en la banda atenuada. Asimismo, se dan las características de la ventana de Kaiser equivalente en cada caso. El parámetro L es la longitud de la ventana (el número de muestras total).

Ventana	Amplitud relativa del lóbulo secundario (dB)	Ancho aproximado del lóbulo principal	Error de aproximación de pico $20\log(\delta)$ (dB)	Ventana de Kaiser equivalente α	Ancho de transición de la ventana de Kaiser equivalente
Rectangular	-13.5	$4\pi/(L+1)$	-21	0	$1.81\pi/L$
Bartlett	-27	$8\pi/L$	-25	1.33	$2.37\pi/L$
Hanning	-31	$8\pi/L$	-44	3.86	$5.01\pi/L$
Hamming	-41	$8\pi/L$	-53	4.86	$6.27\pi/L$
Blackman	-57	$12\pi/L$	-74	7.04	$9.19\pi/M$

tabla 2

5.6. Resumen del método de la ventana para diseñar filtros FIR.

Los pasos a seguir para diseñar un filtro FIR pueden resumirse de la siguiente forma:

Paso 1. Especificar la respuesta en frecuencia ideal o deseada del filtro $H(e^{j\Omega})$.
Paso 2. Obtener la respuesta al impulso $h_d[n]$ del filtro deseado.

Paso 3. Seleccionar una función ventana w[n] que satisfaga las especificaciones de la banda de paso o atenuación, y determinar el número de coeficientes necesarios del filtro.
Paso 4. Obtener los valores de los coeficientes de w[n] y los coeficientes del filtro real : $h[n] = h_d[n] \cdot w[n]$

5.7. Ventajas y desventajas del método de la ventana.

- Una ventaja importante es la simplicidad: La cantidad de cálculos necesarios es muy pequeña.
- La principal desventaja es la falta de flexibilidad.
- Debido al efecto de la convolución del espectro de la función ventana y la respuesta en frecuencia deseada $H_d(e^{j\Omega})$., las frecuencias límite de las bandas de paso y atenuada n o pueden especificarse con precisión.
- Para una ventana dad (excepto para el caso de la función ventana de Kaiser), el rizado máximo de amplitud de la respuesta en frecuencia está fijado independientemente de la longitud de la ventana. Por tanto, la atenuación de la banda atenuada es fija para una cierta ventana. En consecuencia, para una especificación de atenuación, el diseñador debe encontrar la ventana adecuada.
- En algunas aplicaciones, la expresión de $H_d(e^{j\Omega})$. es muy complicada para obtener la respuesta al impulso deseada formula analíticamente, en estos casos formula, puede obtenerse mediante el método de muestreo en frecuencia.

6. Diseño de filtro FIR de fase lineal empleando el método de muestreo en frecuencia.

El método de muestreo en frecuencia permite diseñar filtro FIR o recursivos tanto para filtros estándar (paso bajo, paso alto, etc), como para filtros de respuesta en frecuencia arbitraria. El único atractivo de este método es que también permite realizar filtros FIR recursivos, conduciendo a filtros computacionalmente eficientes.

Con algunas restricciones, pueden diseñarse filtros FIR recursivos cuyos coeficientes son simples número enteros, resultando atractivo cuando sólo son posibles operaciones aritméticas primitivas como en microprocesadores estándar.

En el breve estudio que se va a realizar de este método de filtros FIR solo abordaremos el problema de los filtros no recursivos.

7. **Tipos de filtros**

Los filtros más corrientes son los filtros paso bajo (Low Pass, LP), paso alto (High Pass, HP), paso de banda (Band Pass, BP) y los filtros rechazo de banda (o paso no banda) (Band Reject, Band stop o Notch). En la figura 1 se representan estos 4 tipos de filtros mediante su respuesta en frecuencia o espectro de amplitud.

Cada punto de la respuesta en frecuencia nos indica la atenuación a la que se someterá una señal a una frecuencia determinada.

- Los filtros paso bajo (LP) dejan pasar las frecuencias que están por debajo de una determinada frecuencia.

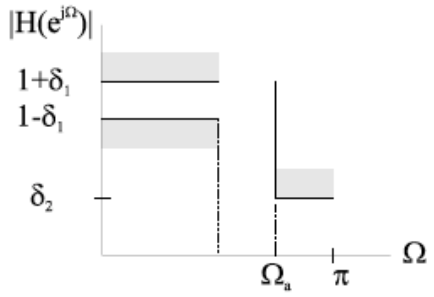


figura 10

- Los filtros paso alto (HP) dejan pasar las frecuencias que están por encima de una determinada frecuencia.

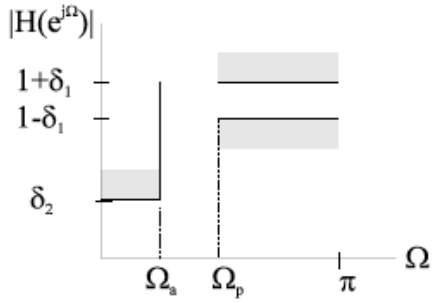


figura 11

- Los filtros paso banda (BP) dejan pasar las frecuencias que están situadas en una determinada banda de frecuencia, es decir, entre dos determinadas frecuencias.

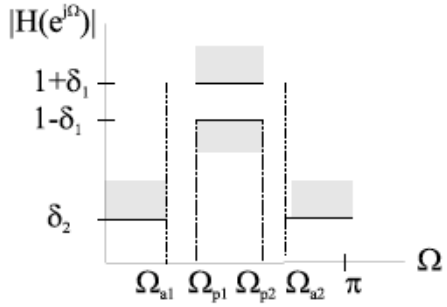


figura 12

- Los filtros rechazo de banda (BR) dejan pasar todas las frecuencias excepto las que están situadas en una determinada banda de frecuencia, es decir, entre dos determinadas frecuencias f1 y f2.

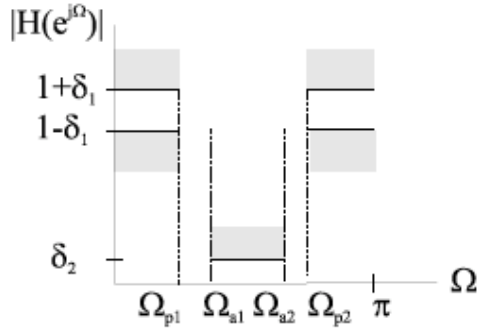


figura 13

8. **<u>Ejemplos del diseño de filtros.</u>**

A la hora de diseñar un filtro digital partir de sus especificaciones podemos centrarnos en un serie de puntos que llegar al correcto diseño del mismo:

1. Calcular la frecuencia de corte. Normalmente se tomará el punto medio de la banda de transición:

$$\omega_c = \frac{\omega_c + \omega_r}{2}$$

2. Calcular el ancho de banda de transición:

$$\Delta_\omega = |\omega_r - \omega_c|$$

3. Elegir la ventana más corta que cumpla las especificaciones de atenuación, para ello se empleará el cuadro resumen de las propiedades de las

ventanas, facilitado anteriormente.
4. A partir de Δ_ω calcular el número de elementos de la ventana.
5. Escribir la respuesta al impulso ideal del tipo de filtro requerido.
6. Multiplicar la respuesta al impulso ideal por la ventuna de truncamiento.

$$h[n] = h_d[n] \cdot w[n]$$

8.1 Ejemplo para el diseño de un filtro paso alto.

Datos: Frecuencia de corte = 24KHz
Frecuencia banda de transición = 3.2KHz
Frecuencia de muestro = 64KHz
Atenuación mínima en la banda atenuada = 50dB
Número máximo de coeficientes del filtro = 100

Paso 1: Relación entre frecuencias $\omega\ y\ f$:

$$\omega = \Omega \cdot T = \frac{2\pi \cdot f}{f_s} \rightarrow f = \frac{\omega \cdot f_s}{2\pi}, donde\ f_s\ es\ la\ frecuencia\ de\ muestreo$$

$$\omega_c = \frac{2\pi \cdot 24}{64} = \frac{3\pi}{4} \qquad \Delta_\omega = \frac{2\pi \cdot 3.2}{64} = \frac{\pi}{10}$$

Paso 2: Elección de la ventana; como las ventana rectángular, Barlet y Hanning producen atenuaciones inferiores a 50dB en la banda atenuada se descartan

- Ventana Hamming: $\Delta_\omega = \frac{8\pi}{M} = \frac{\pi}{10} \rightarrow M = 80\ (81\ coeficientes)$
- Ventana Blackman: $\Delta_\omega = \frac{12\pi}{M} = \frac{\pi}{10} \rightarrow M = 120\ (121\ coeficientes)$, esta ventana se descarta porque no cumple el número máximo de coeficientes del filtro

Paso 3: Respuesta al impulso

$$h[n] = \left[\frac{sen\pi(n-40)}{\pi(n-40)} - \frac{sen\frac{3\pi}{4}(n-40)}{\pi(n-40)} \right] \cdot w_{hanning}[n]$$

8.2 Ejemplo para el diseño de un filtro paso banda.

Datos: Número máximo de muestras: 200
Banda de paso: $\omega_{p1} = \frac{7{,}5\pi}{24}\ y\ \omega_{p1} = \frac{10{,}5\pi}{24}$

Banda atenuada inferior: $0\ a\ \omega_{a1} = \frac{6{,}5\pi}{24}$

Banda atenuada superior: $\omega_{a2} = \frac{11.5\pi}{24}$

Requiriéndose una atenuación mejor de 45dB

Paso 1: Elección de la ventana

Banda de transición inferior: $\omega_{p1} - \omega_{a1} = \frac{7.5\pi}{24} - \frac{6.5\pi}{24} = \frac{\pi}{24}$

Banda de transición superior: $\omega_{a2} - \omega_{p2} = \frac{11.5\pi}{24} - \frac{10.5\pi}{24} = \frac{\pi}{24}$

Como se necesita una atenuación mayor de 45dB las ventanas que cumplen esta característica son la de Hamming y la de Blackman. Se debe escoger la que cumpla la banda de transición de 200 muestras.

Hamming→$\Delta_\omega = \frac{8\pi}{200} = \frac{\pi}{25} < \frac{\pi}{24}$

Elegimos la ventana de Hamming

Blackman→$\Delta_\omega = \frac{12\pi}{200} = \frac{3\pi}{50} > \frac{\pi}{24}$

Paso 2: Respuesta impulsiva del filtro

$$h[n] = \left[\frac{sen(\omega_{p2}(n-100))}{\pi(n-100)} - \frac{sen(\omega_{p1}(n-100)}{\pi(n-100)} + \delta(n-100)\right] \cdot w_{hamming}[n]$$

8.3 Ejemplo para el diseño de un filtro paso bajo

Datos:

El número de coeficientes debe ser lo menor posible

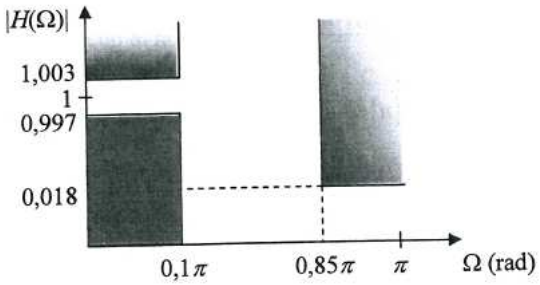


figura 14

Paso 1: Atenuación máxima del filtro y error de aproximación de pico.

$$1-\delta_1 = 0.997 \rightarrow \delta_1 = 0.003 \rightarrow A_1 = -20log\delta_1 = 50.46dB$$
$$\delta_2 = 0.018 \rightarrow A_2 = -20log\delta_2 = 34.9dB$$

El valor más restrictivo es A_1, por lo que solo podríamos usar las ventanas de Hamming o Blackman, puesto que el número de coeficientes debe ser lo menos posible, usaremos la ventana que requiere menos coeficientes para el mismo anda de banda de transición, es decir, la de Hamming.

$$\frac{8\pi}{L} \leq 0.85\pi - 0.1\pi \rightarrow L \geq \frac{8\pi}{0.75\pi} \rightarrow L \geq \frac{32}{3} = 10.67$$

Por lo tanto, necesitaremos al menos 11 coeficientes, lo que supone un filtro de orden M =5 como mínimo.

Para el filtro ideal: $\Omega_c = \frac{0.85\pi+0.1\pi}{2} = 0.475\pi$, con lo que se tiene

$$H\left(e^{j\Omega}\right) = \begin{cases} 1 & |\Omega| \leq \Omega_c \\ 0 & |\Omega| \geq \Omega_c \end{cases} \rightarrow h[n] = \frac{sen(0.475 \cdot \pi n)}{\pi n}$$

Paso 2: Respuesta impulsiva del filtro

$$h[n] = \left[\frac{sen(0.475 \cdot \pi(n-5)}{\pi(n-5)}\right] \cdot w_{hamming}[n]$$

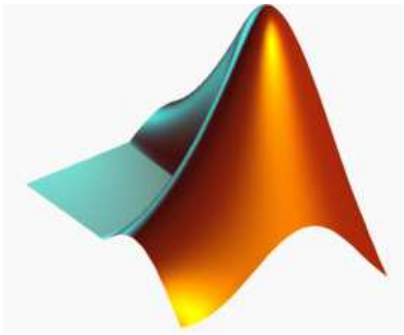
Se desplaza el filtro 5 muestras a la derecha para hacerlo causal

Capítulo III:
La herramienta

1. Introducción

Como se ha venido comentando, el fin último de este Proyecto Fin de Carrera es ayudar al alumno en la autoevaluación de conocimientos, de cara a poder aprobar el examen final del laboratorio. De modo, que el desarrollo de la herramienta ha girado en torno al alumno, a la capacidad de cada de uno de ellos de resolver las práctica y, posteriormente, autoevaluarse a través de la herramienta.

1.1 El entorno de trabajo: Matlab



MATLAB (abreviatura de *MATrix LABoratory*, "laboratorio de matrices") es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).

Fue creado por *Cleve Moler* en 1984, surgiendo la primera versión con la idea de emplear paquetes de subrutinas escritas en Fortran en los cursos de álgebra lineal y análisis numérico, sin necesidad de escribir programas en dicho lenguaje. El lenguaje de programación M fue creado en 1970 para proporcionar un sencillo acceso al software de matrices *LINPACK* y *EISPACK* sin tener que usar Fortran.

Dentro de Matlab, más concretamente, utilizaremos el módulo GUIDE, fue añadido a partir de la versión 5.0 de Matlab. GUIDE (Graphical User Interface Development Environmment) permite crear de modo interactivo la interface de usuario.

A través de GUIDE, se creará la citada interfaz de usuario con la que el alumno interactuará rellenando cada uno de los campos con las soluciones obtenidas de la práctica.

Para cada una de las prácticas se ha realizado tres interfaces distintas:

- ✓ Aprendizaje: Se permite al alumno realizar un número infinito de intentos, indicándole para cada una de las respuestas si es correcta o errónea.
- ✓ Completa: El alumno tiene un máximo de dos intentos por cada pregunta, en cada intento se le indicará si su respuesta es correcta o no.
- ✓ Evaluación: En esta última interfaz el alumno no conoce el resultado de sus respuestas hasta el final, así pues, cuando todas las respuestas han sido introducidas se le indica cuales de ellas son correctas y cuales no lo son.

2. Prácticas de Procesado Digital de la Señal

2.1 Práctica 1: Funciones de ventana

La primera práctica tiene como objetivo analizar las características de las diferentes funciones de ventana que serán empleadas en la asignatura de Procesado Digital de la Señal, tanto en el tema dedicado al diseño de filtros mediante el método de la ventana como en el tema dedicado a las propiedades espectrales de la DFT.

Para la implementación de esta primera práctica a través de GUI´s de Matlab se han creado 5 interfaces de usuario que albergaran todas las preguntas.

2.1.1 Aprendizaje: Primera ventana de petición de resultados

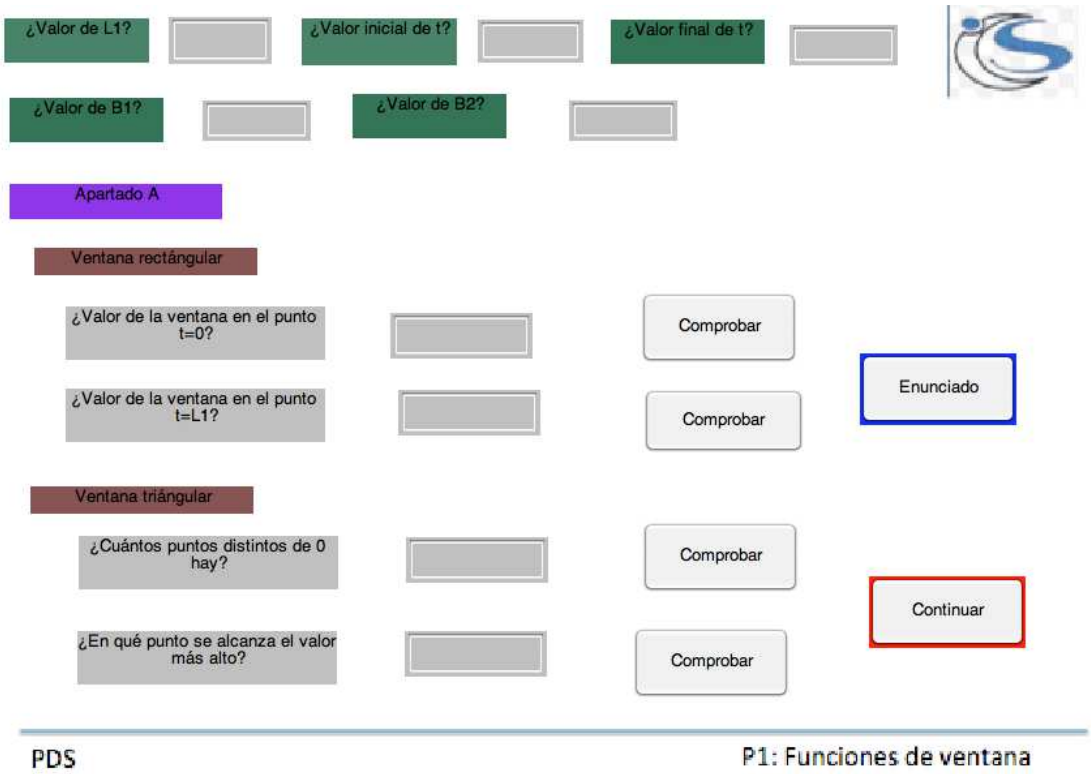


figura 15

En este primer GUI se puede observar una primera parte en la parte superior en que la se piden los datos que se dan en el enunciado de la práctica (existen diferentes enunciados con diferentes datos)
A continuación, se piden los primero resultados obtenidos de resolver la práctica.
En la parte de la izquierda existen dos botones, el primero en el muestra el enunciado de la práctica en formato PDF, y el segundo que nos lleva al

siguiente GUI.

Cuando se pulsa el botón “Comprobar” la respuesta puede ser correcta o incorrecta, las ventanas que aparecerán en cada caso son las siguientes:

figura 15.1

2.1.2 Aprendizaje: Segunda ventana de petición de resultados

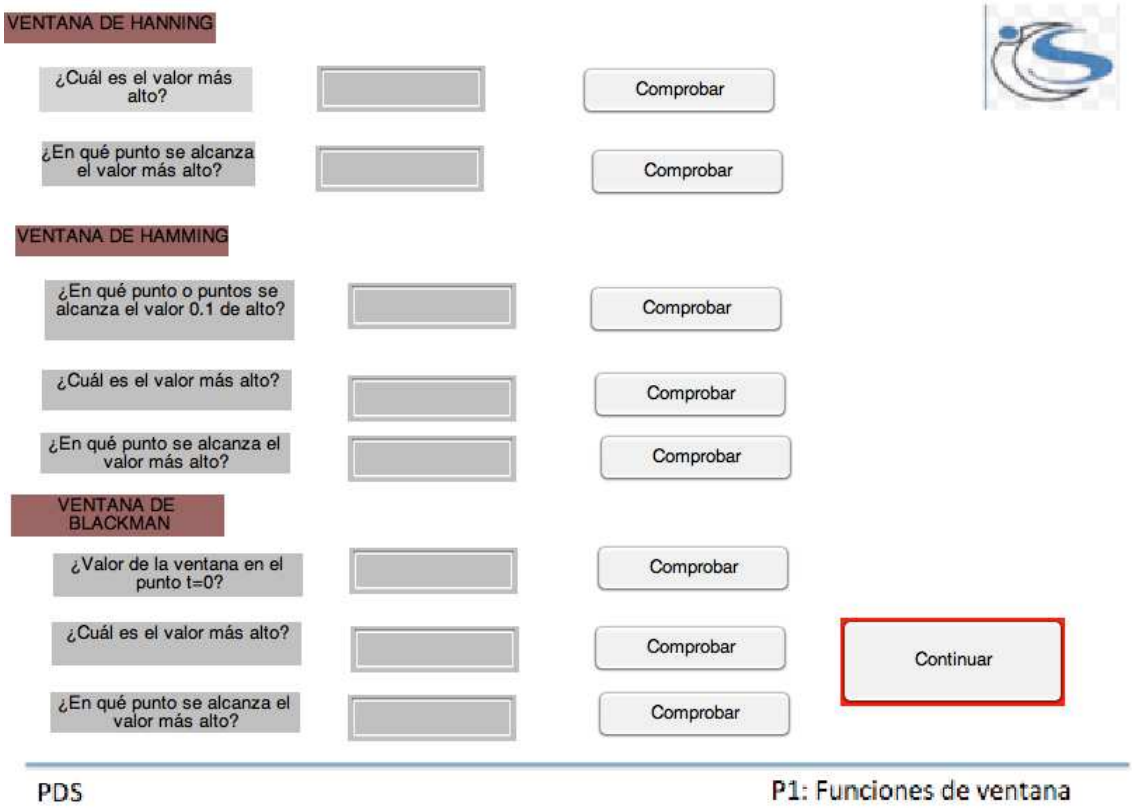


figura 16

En este segundo GUI, se pide al alumno que introduzca valores de las distintas ventanas obtenidas anteriormente.

2.1.3 Aprendizaje: Tercera ventana de petición de resultados

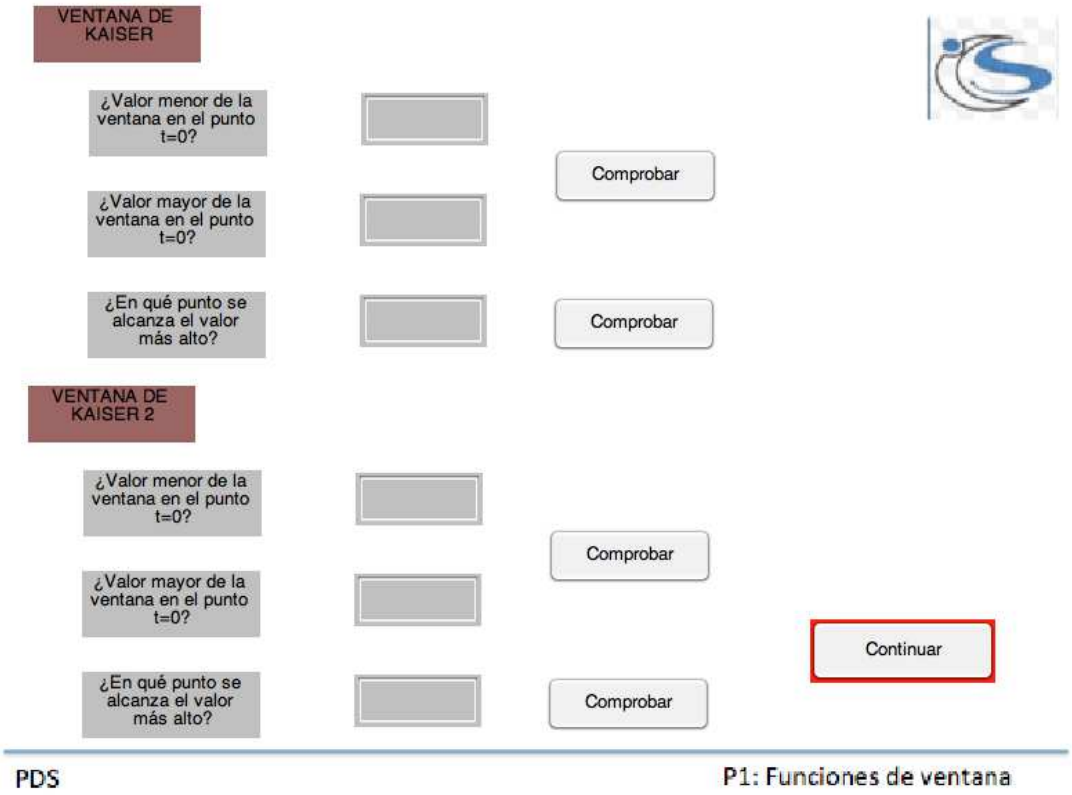


figura 17

En este GUI podemos, al igual que en anterior, se pide al alumno que introduzca los valores obtenidos.

2.1.4 Aprendizaje: Cuarta ventana de petición de resultados

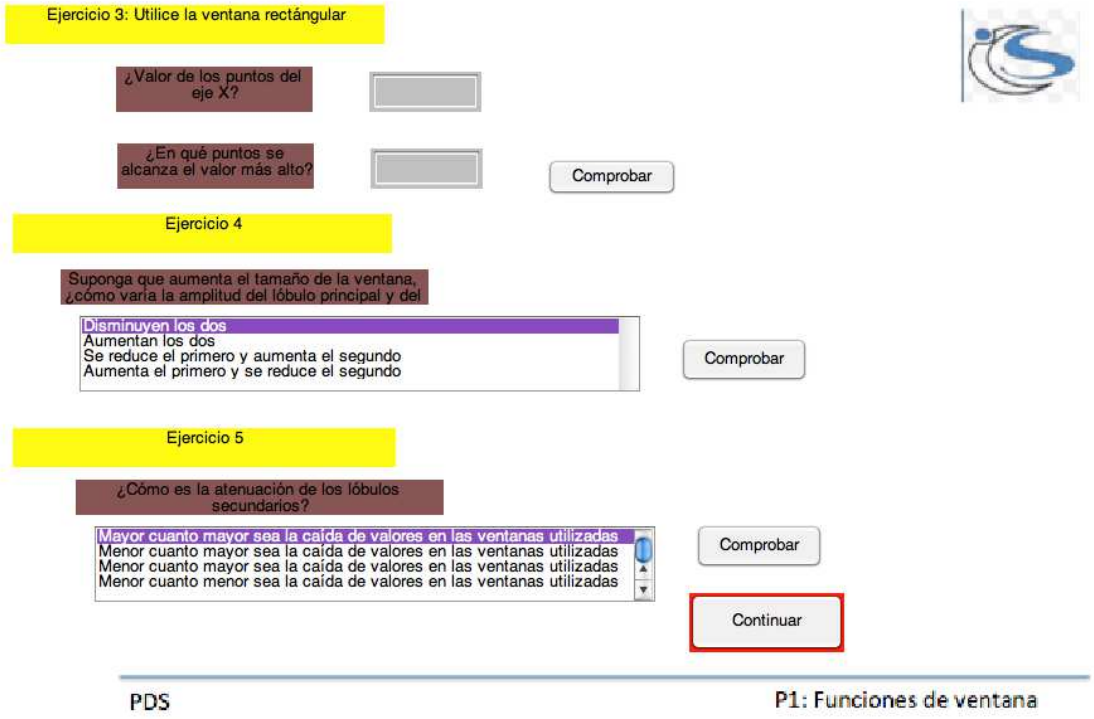


figura 18

En este cuarto GUI, además de pedir valores de tipo número, también se pide la elección de una de las opciones dadas mediante un listbox.

2.1.5 Aprendizaje: Quinta ventana de petición de resultados

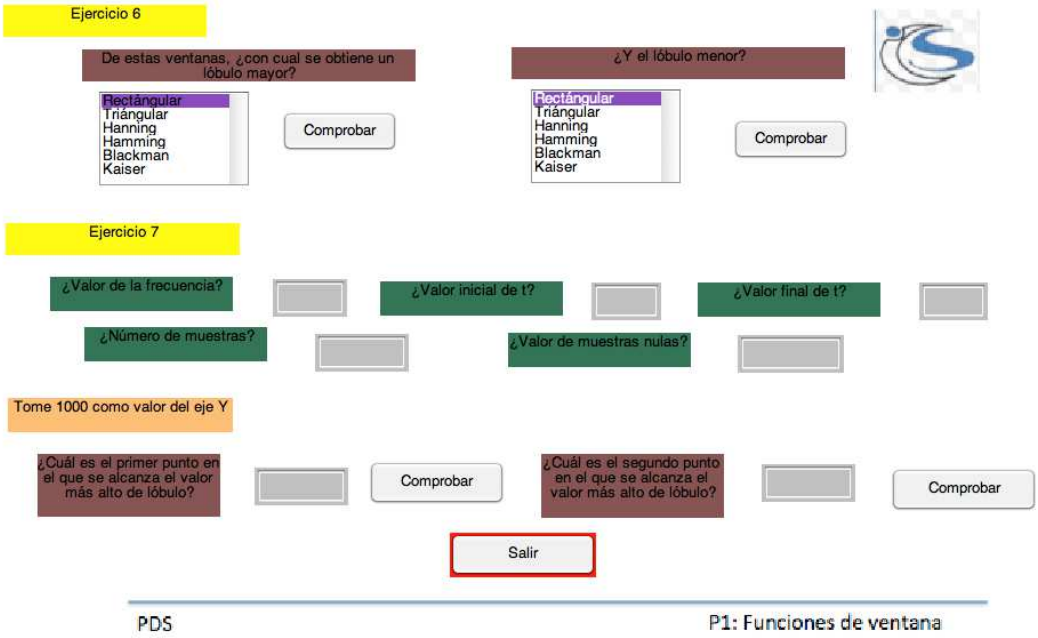


figura 19

En este último GUI, al igual que en el cuarto GUI, se pide al alumno tanto valores númericos como la elección de una de las opciones dadas. Además, por ser el último GUI, también tenemos el botón “Salir”, que cerrará la herramienta. Cuando se pulse este botón aparecerá la siguiente ventana:

figura 20

Para confirmar que se realmente se quiere salir de la aplicación.

2.2 Práctica 2: Diseño de filtros FIR

Esta segunda práctica tiene como objetivo el diseño de filtros digitales de tipo FIR. Se centra en el uso de ventanas utilizadas en la práctica anterior.

Es este caso, se van a mostrar las dos interfaces restantes: Completo y Evaluación.

2.2.1 Completo: Primera ventana de petición de resultados

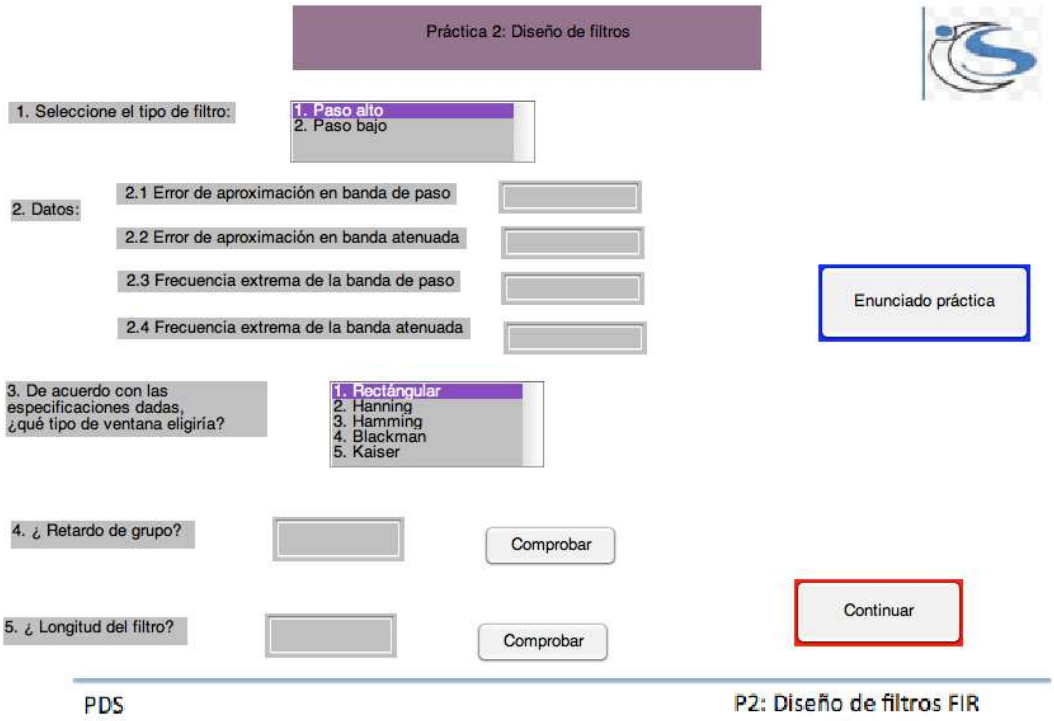


figura 21

En este primer GUI de la segunda práctica se observa que, al principio, se piden los datos facilitados en el enunciado de la práctica, el cual puede abrirse en el botón situado a la izquierda.

A continuación, se pide que se seleccione el tipo de ventana elegida, que debe haber sido calculado anteriormente, y dos datos numéricos para comprobar si son correctos o no lo son.

Las ventanas mostradas como resultado de la comprobación serán distintas dependiendo de si es correcta o no lo es.

figura 21.1

2.2.2 Completo: Segunda ventana de petición de resultados

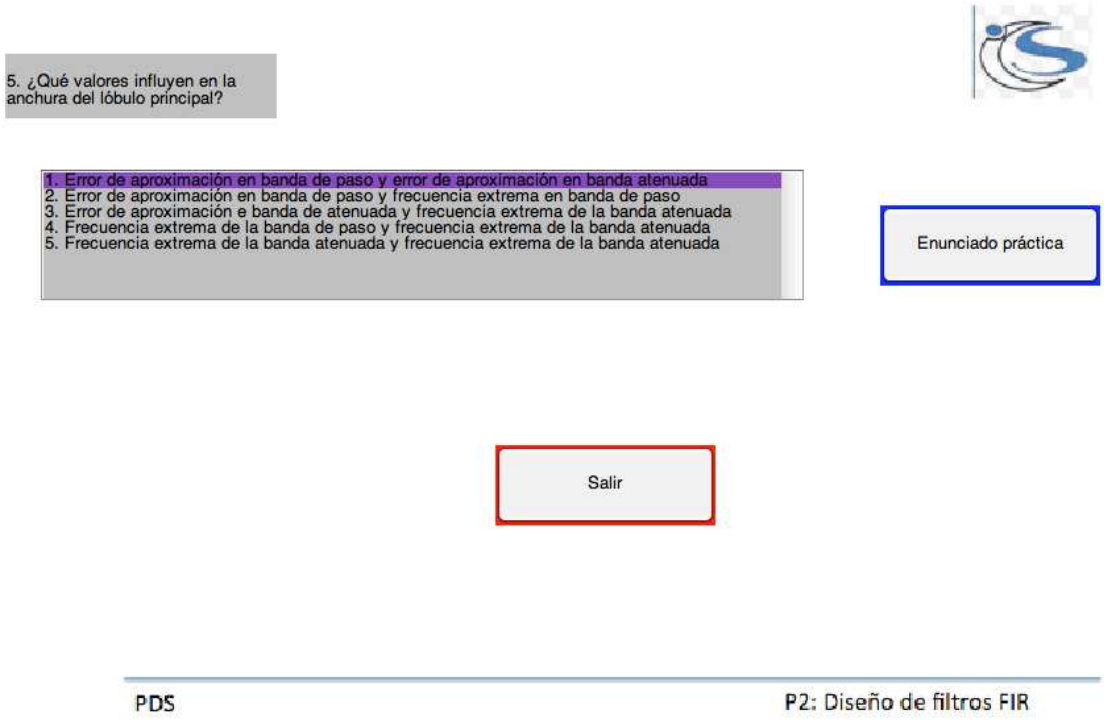


figura 22

En este sgundo y último GUI, se pide al alumno que elija entre una de las opciones dadas en el listbox.

2.2.3 Evaluación: Primera ventana de petición de resultados

En este caso, la interfaz de usuario usada para crear la GUI de tipo evaluación no será diferente de la creada para la GUI de tipo completo o aprendizaje, sin embargo, la diferencia estará en el código.

La GUI de evaluación está pensada para que el alumno no pueda comprobar los resultado hasta el final, en este momento, una ventana aparecerá cuando se pulse el botón “Comprobar” indicándole al alumno cuales de las respuestas eran correctas y cuales erróneas.

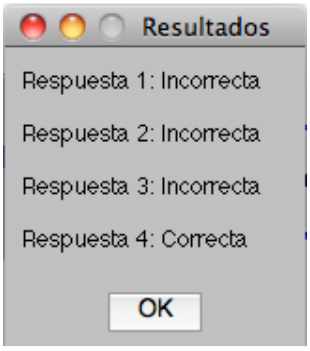


figura 23

3. El alumno

Para la correcta utilización de la herramienta por parte del alumno, se espera que éste haya realizado un estudio previo para ellos, los profesores de la asignatura facilitarán al alumno, junto con el enunciado de la práctica, unos puntos básicos para que no resulte un problema abordar la práctica.

Con estas indicaciones y la resolución del enunciado de la práctica en el entorno Matlab, la utilización de la herramienta hará al alumno más autónomo a la hora de evaluar sus conocimientos.

Capítulo IV: Conclusiones, Anexos y Bibliografía

1. Conclusiones

Una vez acabado este proyecto fin de carrera he llegado a varias conclusiones, tanto a nivel personal como a nivel educativo.

La más importante de ellas es la que a nivel educativo se refiere, con este proyecto se buscaba ahondar en el método e-learning, intentando ofrecer al alumno una herramienta que facilitase su aprendizaje sin tener una dependencia total del profesor, siendo en todo momento consciente de su progreso en la asignatura.

El Campus Virtual con el que cuenta la EUITT es el mejor entorno para lograrlo, a través de la plataforma Moodle se puede llegar al alumno y ofrecerle la herramienta, para que disponga de ella en cualquier momento y lugar, siendo esto, como ya se ha dicho, la base sobre la que se apoya el método e-learning.

Así, se buscaba facilitar tanto al alumno como al profesor la evaluación de los conocimientos adquiridos.

Por lo tanto, una vez acabada la herramienta, se puede afirmar que se ha logrado el objetivo fijado, un entorno de trabajo con una interfaz gráfica sencilla, que el alumno podrá usar sin dificultades.

A nivel personal, el proyecto ha significado un agradable reto, la utilización de Matlab durante la carrera siempre estuvo ligada a la parte matemática que ofrece, para enfrentarme al proyecto, tuve que descubrir los aspectos de Matlab como entorno de programación y más aún, como diseñador de interfaces gráficas.

Matlab ofrece multitud posibilidades desconocidas por la mayoría de los usuarios, el haber descubierto esta cara desconocida de Matlab ha supuesto para mi muchas horas de trabajo y una agradable recompensa.

Por todo lo anterior, puedo decir que el proyecto ha cumplido con las expectativas iniciales y ha conseguido su objetivo: la creación de una herramienta que permita al alumno autocorregir sus prácticas.

2. Continuación del proyecto

Este proyecto fin de carrera ofrece dos vías de continuación:

i. Realización de la herramienta que solucione el resto de prácticas, el proyecto se centra en la resolución de las dos primeras prácticas de la asignatura, constando ésta de otras 4, que podrían ser creadas en otros proyectos.
ii. Integración de la herramienta en la plataforma Moodle, para conseguir que esté accesible por el alumno de forma sencilla.

3. **Código de la herramienta.**

3.1 Práctica 1: Funciones de ventana

3.1.1 Aprendizaje: Código primera ventana de petición de resultados

```
function varargout = gui1(varargin)
% GUI1 MATLAB code for gui1.fig
%      GUI1, by itself, creates a new GUI1 or raises the existing
%      singleton*.
%
%      H = GUI1 returns the handle to a new GUI1 or the handle to
%      the existing singleton*.
%
%      GUI1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI1.M with the given input arguments.
%
%      GUI1('Property','Value',...) creates a new GUI1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before gui1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to gui1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui1

% Last Modified by GUIDE v2.5 29-Jun-2013 21:41:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui1_OpeningFcn, ...
                   'gui_OutputFcn',  @gui1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before gui1 is made visible.
function gui1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui1 (see VARARGIN)

% Choose default command line output for gui1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
background=imread('imagen.jpg');%imagen de fondo
axes(handles.background);

axis off;

imshow(background);
handles.output=hObject;
guidata(hObject,handles);


% --- Outputs from this function are returned to the command line.
function varargout = gui1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
Val = get(hObject, 'String');
NewVal = str2double(Val);
handles.edit2 = NewVal;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double
Val2 = get(hObject, 'String');
NewVal2 = str2double(Val2);
handles.edit3 = NewVal2;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double
Val3 = get(hObject, 'String');
NewVal3 = str2double(Val3);
handles.edit4 = NewVal3;
guidata(hObject, handles);
function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double
Val4 = get(hObject, 'String');
NewVal4 = str2double(Val4);
handles.edit7 = NewVal4;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double
Val5 = get(hObject, 'String');
NewVal5 = str2double(Val5);
handles.edit8 = NewVal5;
guidata(hObject, handles);
```

```
% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

t = newVal2:newVal3;

function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double
%Primer valor ventana rect·ngular
Val6 = get(hObject, 'String');
NewVal6 = str2double(Val6);
handles.edit9 = NewVal6;
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%----------RECTÁNGULAR--------------
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if (handles.edit9==1)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(handles.edit9,'String',ini);
else
    msgbox('Respuesta incorrecta','Comprobar');
end
function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double
Val7 = get(hObject, 'String');
NewVal7 = str2double(Val7);
handles.edit10 = NewVal7;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (handles.edit10==0)
   msgbox('Respuesta correcta','Comprobar');
   ini=char(' ');
   set(handles.edit10,'String',ini);
else
   msgbox('Respuesta incorrecta','Comprobar');

end


function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a double
Val8 = get(hObject, 'String');
NewVal8 = str2double(Val8);
handles.edit11 = NewVal8;
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


%--------TRIÁNGULAR-----------------
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

if (handles.edit11==handles.edit2-2)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(handles.edit11,'String',ini);
else
   msgbox('Respuesta incorrecta','Comprobar');

end


function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12 as a double
Val9 = get(hObject, 'String');
NewVal9 = str2double(Val9);
handles.edit12 = NewVal9;
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (handles.edit12==handles.edit2/2||handles.edit12==handles.edit2/2+1)
   msgbox('Respuesta correcta','Comprobar');
   ini=char(' ');
   set(handles.edit12,'String',ini);
```

```
else
   msgbox('Respuesta incorrecta.','Comprobar');

end


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
L1 = handles.edit2;
tinicial = handles.edit3;
tfinal = handles.edit4;
b1 = handles.edit7;
b2 = handles.edit8;

save variables L1 tinicial tfinal b1 b2

botok=uicontrol('Style','pushbutton','Units','normalized','Position',[.84 .03 .12
.05],'String','CONTINUAR','Callback','clear all; close all;clc; gui2;');
```

3.1.2 Aprendizaje: Código segunda ventana de petición de resultados

```
unction varargout = gui2(varargin)
%GUI2 M-file for gui2.fig
%      GUI2, by itself, creates a new GUI2 or raises the existing
%      singleton*.
%
%      H = GUI2 returns the handle to a new GUI2 or the handle to
%      the existing singleton*.
%
%      GUI2('Property','Value',...) creates a new GUI2 using the
%      given property value pairs. Unrecognized properties are passed via
%      varargin to gui2_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      GUI2('CALLBACK') and GUI2('CALLBACK',hObject,...) call the
%      local function named CALLBACK in GUI2.M with the given input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

```
% Edit the above text to modify the response to help gui2

% Last Modified by GUIDE v2.5 11-Jun-2013 20:16:34

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui2_OpeningFcn, ...
                   'gui_OutputFcn',  @gui2_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gui2 is made visible.
function gui2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% Choose default command line output for gui2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

background=imread('imagen.jpg');%imagen de fondo
axes(handles.background);

axis off;
```

```
imshow(background);
handles.output=hObject;
guidata(hObject,handles);


% --- Outputs from this function are returned to the command line.
function varargout = gui2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13 as a double
Val10 = get(hObject, 'String');
NewVal10 = str2double(Val10);
handles.edit13 = NewVal10;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

%-----------HANNING------------
% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
if (handles.edit13==1)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(handles.edit13,'String',ini);
else
    msgbox('Respuesta incorrecta.','Comprobar');
end
function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14 as a double
Val11 = get(hObject, 'String');
NewVal11 = str2double(Val11);
handles.edit14 = NewVal11;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

load variables;
if (handles.edit14==L1/2||handles.edit14==L1/2+1)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(handles.edit14,'String',ini);
else
    msgbox('Respuesta incorrecta.','Comprobar');
end
```

```
 function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15 as a double

Val12 = get(hObject, 'String');
NewVal12 = str2double(Val12);
handles.edit15 = NewVal12;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%------------HAMMING--------------
% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

load variables;

if (handles.edit15==2||handles.edit12==L1-1)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(handles.edit15,'String',ini);
else
    msgbox('Respuesta incorrecta','Comprobar');
end

function edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit16 as text
%        str2double(get(hObject,'String')) returns contents of edit16 as a double
Val13 = get(hObject, 'String');
NewVal13 = str2double(Val13);
handles.edit16 = NewVal13;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (handles.edit16==1)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(handles.edit16,'String',ini);
else
    msgbox('Respuesta incorrecta','Comprobar');
end

function edit17_Callback(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit17 as text
%        str2double(get(hObject,'String')) returns contents of edit17 as a double
Val14 = get(hObject, 'String');
NewVal14 = str2double(Val14);
handles.edit17 = NewVal14;
guidata(hObject, handles);
```

```
% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton13.
function pushbutton13_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load variables;

if (handles.edit17==L1/2||handles.edit17==L1/2+1)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(handles.edit17,'String',ini);
else
    msgbox('Respuesta incorrecta.','Comprobar');
end

function edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit18 as text
%        str2double(get(hObject,'String')) returns contents of edit18 as a double
Val15 = get(hObject, 'String');
NewVal15 = str2double(Val15);
handles.edit18 = NewVal15;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


%---------BLACKMAN--------------
% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (handles.edit18==0)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(handles.edit18,'String',ini);
else
    msgbox('Respuesta incorrecta','Comprobar');

end

function edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit19 as text
%        str2double(get(hObject,'String')) returns contents of edit19 as a double
Val16 = get(hObject, 'String');
NewVal16 = str2double(Val16);
handles.edit19 = NewVal16;
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in pushbutton15.
function pushbutton15_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (handles.edit19==1)
   msgbox('Respuesta correcta','Comprobar');
   ini=char(' ');
   set(handles.edit19,'String',ini);
else
   msgbox('Respuesta incorrecta','Comprobar');

end


function edit20_Callback(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit20 as text
%        str2double(get(hObject,'String')) returns contents of edit20 as a double
Val17 = get(hObject, 'String');
NewVal17 = str2double(Val17);
handles.edit20 = NewVal17;
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton16.
function pushbutton16_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

load variables;

if (handles.edit20==L1/2||handles.edit20==L1/2+1)
   msgbox('Respuesta correcta','Comprobar');
   ini=char(' ');
   set(handles.edit20,'String',ini);
else
   msgbox('Respuesta incorrecta.','Comprobar');

end

% --- Executes on button press in pushbutton17.
function pushbutton17_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
botok=uicontrol('Style','pushbutton','Units','normalized','Position',[.84 .03 .12
.05],'String','CONTINUAR','Callback','clear all; close all;clc; gui3;');
```

3.1.3 Aprendizaje: Código tercera ventana de petición de resultados

```
function varargout = gui3(varargin)
% GUI3 MATLAB code for gui3.fig
%      GUI3, by itself, creates a new GUI3 or raises the existing
%      singleton*.
%
%      H = GUI3 returns the handle to a new GUI3 or the handle to
%      the existing singleton*.
%
%      GUI3('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI3.M with the given input arguments.
%
%      GUI3('Property','Value',...) creates a new GUI3 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before gui3_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to gui3_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

```
% Edit the above text to modify the response to help gui3

% Last Modified by GUIDE v2.5 29-Jun-2013 18:25:24

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui3_OpeningFcn, ...
                   'gui_OutputFcn',  @gui3_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);

if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gui3 is made visible.
function gui3_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui3 (see VARARGIN)

% Choose default command line output for gui3
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui3 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
background=imread('imagen.jpg');%imagen de fondo
axes(handles.background);

axis off;
```

```
imshow(background);
handles.output=hObject;
guidata(hObject,handles);
% --- Outputs from this function are returned to the command line.
function varargout = gui3_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
Val = get(hObject, 'String');
NewVal = str2double(Val);
handles.edit1 = NewVal;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
Val1 = get(hObject, 'String');
NewVal1 = str2double(Val1);
handles.edit2 = NewVal1;
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


%------------KAISER1------------
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (handles.edit1==0.4&&handles.edit2==0.5)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(handles.edit1,'String',ini);
    set(handles.edit2,'String',ini);
else
    if(handles.edit1~=0.4&&handles.edit2==0.5)
        msgbox('Valor menor incorrecto','Comprobar');
        ini=char(' ');
        set(handles.edit1,'String',ini);
        set(handles.edit2,'String',ini);

    elseif(handles.edit1==0.4&&handles.edit2~=0.5)
        msgbox('Valor mayor incorrecto.','Comprobar');
        ini=char(' ');
        set(handles.edit1,'String',ini);
        set(handles.edit2,'String',ini);
```

```
    else
        msgbox('Ambos valores son incorrectos.','Comprobar');
    end
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double
Val3 = get(hObject, 'String');
NewVal3 = str2double(Val3);
handles.edit3 = NewVal3;
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load variables;

if (handles.edit3==L1/2||handles.edit3==L1/2-1)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(handles.edit3,'String',ini);
else
    msgbox('Respuesta incorrecta.','Comprobar');

end
```

```
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double
Val4 = get(hObject, 'String');
NewVal4 = str2double(Val4);
handles.edit4 = NewVal4;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called



% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double
Val5 = get(hObject, 'String');
NewVal5 = str2double(Val5);
handles.edit5 = NewVal5;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%--------------KAISER2-----------
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if (handles.edit4==0.4&&handles.edit5==0.5)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(handles.edit4,'String',ini);
    set(handles.edit5,'String',ini);
else
    if(handles.edit4~=0.4&&handles.edit5==0.5)
        msgbox('Valor menor incorrecto','Comprobar');
        ini=char(' ');
        set(handles.edit4,'String',ini);
        set(handles.edit5,'String',ini);
    elseif(handles.edit4==0.4&&handles.edit5~=0.5)
        msgbox('Valor mayor incorrecto.','Comprobar');
        ini=char(' ');
        set(handles.edit4,'String',ini);
        set(handles.edit4,'String',ini);
     else
        msgbox('Ambos valores son incorrectos. Queda un intento','Comprobar');
    end

end
function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double
Val6 = get(hObject, 'String');
NewVal6 = str2double(Val6);
handles.edit6 = NewVal6;
guidata(hObject, handles);
```

```
% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

load variables;

if (handles.edit6==L1/2||handles.edit6==L1/2-1)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(handles.edit6,'String',ini);
else
    msgbox('Respuesta incorrecta.','Comprobar');

end
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
botok=uicontrol('Style','pushbutton','Units','normalized','Position',[.84 .03 .12
.05],'String','CONTINUAR','Callback','clear all; close all;clc; gui10;');
```

3.1.4 Aprendizaje: Código cuarta ventana de petición de resultados

```
function varargout = gui10(varargin)
% GUI10 MATLAB code for gui10.fig
%      GUI10, by itself, creates a new GUI10 or raises the existing
%      singleton*.
%
%      H = GUI10 returns the handle to a new GUI10 or the handle to
```

```
%      the existing singleton*.
%
%      GUI10('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI10.M with the given input arguments.
%
%      GUI10('Property','Value',...) creates a new GUI10 or raises the
%      existing singleton*.  Starting from the left, property value pairs %      are
%      applied to the GUI before gui10_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property %application
%      stop.  All inputs are passed to gui10_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only %one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui10

% Last Modified by GUIDE v2.5 17-Jul-2013 18:41:18

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui10_OpeningFcn, ...
                   'gui_OutputFcn',  @gui10_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gui10 is made visible.
function gui10_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to gui10 (see VARARGIN)

% Choose default command line output for gui10
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui10 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
background=imread('imagen.jpg');%imagen de fondo
axes(handles.background);

axis off;

imshow(background);
handles.output=hObject;
guidata(hObject,handles);


% --- Outputs from this function are returned to the command line.
function varargout = gui10_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
function background_CreateFcn(hObject, eventdata, handles)
% hObject    handle to background (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate background



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
Val = get(hObject, 'String');
NewVal = str2double(Val);
handles.edit1 = NewVal;
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
Val1 = get(hObject, 'String');
NewVal1 = str2double(Val1);
handles.edit2 = NewVal1;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
%Ejercicio 3
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (handles.edit2==0||handles.edit2==handles.edit1)
   msgbox('Respuesta correcta','Comprobar');
   ini=char(' ');
   set(handles.edit2,'String',ini);
else
   msgbox('Respuesta incorrecta','Comprobar');

end

%Ejercicio 4
% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1 contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from listbox1
% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
indice_seleccionado = get(handles.listbox1,'Value');
if (indice_seleccionado==2)
```

```
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(indice_seleccionado,'String',ini);
else
    msgbox('Respuesta incorrecta.','Comprobar');

end

% --- Executes on selection change in listbox2.
function listbox2_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox2 contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from listbox2

%Ejercicio 5
% --- Executes during object creation, after setting all properties.
function listbox2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
indice_seleccionado = get(handles.listbox2,'Value');

if (indice_seleccionado==1)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(indice_seleccionado,'String',ini);
else
    msgbox('Respuesta incorrecta.','Comprobar');
 end
```

```
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

botok=uicontrol('Style','pushbutton','Units','normalized','Position',[.84 .03 .12
.05],'String','CONTINUAR','Callback','clear all; close all;clc; gui11;');
```

3.1.5 Aprendizaje: Código quinta ventana de petición de resultados

```
function varargout = gui11(varargin)
% GUI11 MATLAB code for gui11.fig
%      GUI11, by itself, creates a new GUI11 or raises the existing
%      singleton*.
%
%      H = GUI11 returns the handle to a new GUI11 or the handle to
%      the existing singleton*.
%
%      GUI11('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI11.M with the given input arguments.
%
%      GUI11('Property','Value',...) creates a new GUI11 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before gui11_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to gui11_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui11

% Last Modified by GUIDE v2.5 08-Jul-2013 19:52:18

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui11_OpeningFcn, ...
                   'gui_OutputFcn',  @gui11_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
```

```
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gui11 is made visible.
function gui11_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui11 (see VARARGIN)

% Choose default command line output for gui11
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui11 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
background=imread('imagen.jpg');%imagen de fondo
axes(handles.background);
axis off;

imshow(background);
handles.output=hObject;
guidata(hObject,handles);


% --- Outputs from this function are returned to the command line.
function varargout = gui11_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1 contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from listbox1


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
indice_seleccionado = get(handles.listbox1,'Value');

if (indice_seleccionado==1)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(indice_seleccionado,'String',ini);
else
    msgbox('Respuesta incorrecta.','Comprobar');

end


% --- Executes on selection change in listbox2.
function listbox2_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox2 contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from listbox2


% --- Executes during object creation, after setting all properties.
function listbox2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
indice_seleccionado = get(handles.listbox2,'Value');

if (indice_seleccionado==5)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(indice_seleccionado,'String',ini);
else
    msgbox('Respuesta incorrecta.','Comprobar');
end


function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
Val = get(hObject, 'String');
```

```
NewVal = str2double(Val);
handles.edit1 = NewVal;
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
Val1 = get(hObject, 'String');
NewVal1 = str2double(Val1);
handles.edit2 = NewVal1;
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double
Val2 = get(hObject, 'String');
NewVal2 = str2double(Val2);
handles.edit3 = NewVal2;
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double
Val3 = get(hObject, 'String');
NewVal3 = str2double(Val3);
handles.edit4 = NewVal3;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double
Val4 = get(hObject, 'String');
NewVal4 = str2double(Val4);
handles.edit5 = NewVal4;
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double
Val5 = get(hObject, 'String');
NewVal5 = str2double(Val5);
handles.edit7 = NewVal5;
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
```

```
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if (handles.edit7==21)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(handles.edit7,'String',ini);
else
    msgbox('Respuesta incorrecta.','Comprobar');

end
function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double
Val6 = get(hObject, 'String');
NewVal6 = str2double(Val6);
handles.edit8 = NewVal6;
guidata(hObject, handles);
 % --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

if (handles.edit8==981)
    msgbox('Respuesta correcta','Comprobar');
    ini=char(' ');
    set(handles.edit8,'String',ini);
else
    msgbox('Respuesta incorrecta','Comprobar');

end

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ans=questdlg('¿Desea salir del programa?','SALIR','Si','No','No');
if strcmp(ans,'No')
return;
end
     clear,clc,close all
```

3.2 Práctica 2: Diseño de filtros FIR

3.2.1 Completo: Código de la primera ventana de petición de resultados

```
function varargout = gui1(varargin)
% GUI1 MATLAB code for gui1.fig
%      GUI1, by itself, creates a new GUI1 or raises the existing
%      singleton*.
%
%      H = GUI1 returns the handle to a new GUI1 or the handle to
%      the existing singleton*.
%
%      GUI1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI1.M with the given input arguments.
%
%      GUI1('Property','Value',...) creates a new GUI1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before gui1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to gui1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui1

% Last Modified by GUIDE v2.5 31-Jul-2013 19:08:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui1_OpeningFcn, ...
                   'gui_OutputFcn',  @gui1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT


% --- Executes just before gui1 is made visible.
function gui1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui1 (see VARARGIN)

% Choose default command line output for gui1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

background=imread('imagen2.jpg');%imagen de fondo
axes(handles.background);

axis off;

imshow(background);
handles.output=hObject;
guidata(hObject,handles);
global contador contador1 contador2;
contador = 0;
contador1 = 0;
contador2 = 0;


% UIWAIT makes gui1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = gui1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
open('practica2.pdf');

% --- Executes on button press in listbox2.
function listbox2_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of listbox2
global indice_seleccionado1;
indice_seleccionado1 = get(handles.listbox2,'Value');

% --- Executes during object creation, after setting all properties.
function listbox2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1 contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from listbox1

global indice_seleccionado1;
global M;
global L;
global contador;

indice_seleccionado2 = get(handles.listbox1,'Value');

if(handles.edit1<handles.edit2)
   Aa = -20*log10(handles.edit1);
else
   Aa = -20*log10(handles.edit2);
end
```

```
if(indice_seleccionado1==1)%Filtro paso alto
    valor = handles.edit3-handles.edit4;
    switch Aa
    case 1,
        Aa <= 21;
        %rect·ngular-->rectwin(L)
        M = ceil(4*pi/valor)-1;
        ventana = 1;
    case 2,
        Aa <= 44;
        %hanning-->hanning(L)
        M = ceil(8*pi/valor);
        ventana = 2;
    case 3,
        Aa <= 53;
        %hamming-->hamming(L)
        M = ceil(8*pi/valor);
        ventana = 3;
    case 4,
        Aa <= 74;
        %blackman-->blackman(L)
        M = ceil(8*pi/valor);
        ventana = 4;
    otherwise
        Aa>74;
        %kaiser-->kaiser(2*M+1,alfa)
        if(Aa<=21)
            beta = 0;
        elseif(Aa>=50)
            beta = 0.1102*(Aa-8.7);
        else
            beta = 0.5842*(Aa-21)^.4+0.07886*(Aa-21);
        end

        if(Aa<=21)
            M =ceil(5.794/valor);
        else
            M =ceil((Aa-7.95)/(28.72*valor));
        end
        ventana = 5;
    end

else
    %Filtro paso bajo
    wc = handles.edit3 + (handles.edit4+handles.edit3)/2;
    valor = handles.edit4-handles.edit3
```

```
 switch Aa
   case 1,
      Aa <= 21;
      %rect·ngular-->rectwin(L)
      M = ceil(4*pi/valor)-1;
      ventana = 1;
   case 2,
      Aa <= 44;
      %hanning-->hanning(L)
      M = ceil(8*pi/valor);
      ventana = 2;
   case 3,
      Aa <= 53;
      %hamming-->hamming(L)
      M = ceil(8*pi/valor);
      ventana = 3;
   case 4,
      Aa <= 74;
      %blackman-->blackman(L)
      M = ceil(8*pi/valor);
      ventana = 4;
   otherwise
     Aa>74;
     %kaiser-->kaiser(2*M+1,alfa)
     if(Aa<=21)
        beta = 0;
     elseif(Aa>=50)
        beta = 0.1102*(Aa-8.7);
     else
        beta = 0.5842*(Aa-21)^.4+0.07886*(Aa-21);
     end

     if(Aa<=21)
        M =ceil(5.794/valor);
     else
        M =ceil((Aa-7.95)/(28.72*valor));
     end
     ventana = 5;
   end

end

L = M*2+1;

if (contador == 0 || contador == 1)
   if (indice_seleccionado2==ventana)
```

```
        contador = 2;
        msgbox('Respuesta correcta','Comprobar');
    else
        if(contador==0)
            msgbox('Respuesta incorrecta. Queda un intento','Comprobar');
            contador = contador+1;
        else
            msgbox('Respuesta incorrecta. No quedan m·s intentos','Comprobar');
            contador = contador + 1;
        end
    end
else
  msgbox('No quedan intentos','Comprobar');

end

% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


%Error de aproximaciÛn en banda de paso
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
Val = get(hObject, 'String');
NewVal = str2double(Val);
handles.edit1 = NewVal;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


%Error de aproximación en banda atenuada
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
Val1 = get(hObject, 'String');
NewVal1 = str2double(Val1);
handles.edit2 = NewVal1;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

%Frecuencia extrema de la banda de paso
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double
Val2 = get(hObject, 'String');
```

```
NewVal2 = str2double(Val2);
handles.edit3 = NewVal2;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%Frecuencia extrema de la banda atenuada
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double
Val3 = get(hObject, 'String');
NewVal3 = str2double(Val3);
handles.edit4 = NewVal3;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double
Val4 = get(hObject, 'String');
NewVal4 = str2double(Val4);
handles.edit6 = NewVal4;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global M;
global contador1;

if (contador1 == 0 || contador1 == 1)
    if (handles.edit6==M)
        contador1 = 2;
        msgbox('Respuesta correcta','Comprobar');
        ini=char(' ');
        set(handles.edit6,'String',ini);
    else
        if(contador1==0)
            msgbox('Respuesta incorrecta. Queda un intento','Comprobar');
            contador1 = contador1+1;
        else
            msgbox('Respuesta incorrecta. No quedan m·s intentos','Comprobar');
            contador1 = contador1 + 1;
        end

    end
```

```
else
  msgbox('No quedan intentos','Comprobar');
end


function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double
Val5 = get(hObject, 'String');
NewVal5 = str2double(Val5);
handles.edit7 = NewVal5;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global L;
global contador2;

if (contador2 == 0 || contador2 == 1)
   if (handles.edit7==L)
      contador2 = 2;
      msgbox('Respuesta correcta','Comprobar');
      ini=char(' ');
      set(handles.edit7,'String',ini);
   else
```

```
        if(contador2==0)
           msgbox('Respuesta incorrecta. Queda un intento','Comprobar');
           contador2 = contador2+1;
        else
           msgbox('Respuesta incorrecta. No quedan m·s intentos','Comprobar');
           contador2 = contador2 + 1;
        end

    end

else
  msgbox('No quedan intentos','Comprobar');
end


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


botok=uicontrol('Style','pushbutton','Units','normalized','Position',[.84 .03 .12
.05],'String','CONTINUAR','Callback','clear all; close all;clc; gui2;');
```

3.2.2 Completo: Código de la segunda ventana de petición de resultados

```
function varargout = gui2(varargin)
% GUI2 MATLAB code for gui2.fig
%      GUI2, by itself, creates a new GUI2 or raises the existing
%      singleton*.
%
%      H = GUI2 returns the handle to a new GUI2 or the handle to
%      the existing singleton*.
%
%      GUI2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI2.M with the given input arguments.
%
%      GUI2('Property','Value',...) creates a new GUI2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before gui2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to gui2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
```

```
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui2

% Last Modified by GUIDE v2.5 19-Aug-2013 18:29:52

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui2_OpeningFcn, ...
                   'gui_OutputFcn',  @gui2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gui2 is made visible.
function gui2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui2 (see VARARGIN)

% Choose default command line output for gui2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

background=imread('imagen2.jpg');%imagen de fondo
axes(handles.background);

axis off;
```

```
imshow(background);
handles.output=hObject;
guidata(hObject,handles);
global contador;
contador = 0;

% UIWAIT makes gui2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = gui2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1 contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from listbox1

global contador;
indice_seleccionado3 = get(handles.listbox1,'Value');

if (contador == 0 || contador == 1)
   if (indice_seleccionado3==4)
      contador = 2;
      msgbox('Respuesta correcta','Comprobar');
      ini=char(' ');
      set(indice_seleccionado3,'String',ini);
   else
      if(contador==0)
         msgbox('Respuesta incorrecta. Queda un intento','Comprobar');
         contador = contador+1;
      else
         msgbox('Respuesta incorrecta. No quedan m·s intentos','Comprobar');
         contador = contador + 1;
```

```
        end

    end

else
  msgbox('No quedan intentos','Comprobar');
end

% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
open('practica2.pdf');


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ans=questdlg('¿Desea salir del programa?','SALIR','Si','No','No');
if strcmp(ans,'No')
return;
end
clear,clc,close all
```

3.2.3 Evaluación: Código de la primera ventana de petición de resultados

```
function varargout = gui1(varargin)
% GUI1 MATLAB code for gui1.fig
%      GUI1, by itself, creates a new GUI1 or raises the existing
%      singleton*.
%
%      H = GUI1 returns the handle to a new GUI1 or the handle to
%      the existing singleton*.
%
%      GUI1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI1.M with the given input arguments.
%
%      GUI1('Property','Value',...) creates a new GUI1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before gui1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to gui1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui1

% Last Modified by GUIDE v2.5 31-Jul-2013 19:08:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui1_OpeningFcn, ...
                   'gui_OutputFcn',  @gui1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before gui1 is made visible.
function gui1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui1 (see VARARGIN)

% Choose default command line output for gui1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

background=imread('imagen2.jpg');%imagen de fondo
axes(handles.background);

axis off;

imshow(background);
handles.output=hObject;
guidata(hObject,handles);
global L;
global M;
global indice_seleccionado1;
global indice_seleccionado2;
global ventana;
indice_seleccionado1 = 0;
M = 0;
L = 0;
indice_seleccionado2 = 0;
ventana = 0;


% UIWAIT makes gui1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = gui1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton2.
```

```
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
open('practica2.pdf');

% --- Executes on button press in listbox2.
function listbox2_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of listbox2
global indice_seleccionado1;
indice_seleccionado1 = get(handles.listbox2,'Value');

% --- Executes during object creation, after setting all properties.
function listbox2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1 contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from listbox1

global indice_seleccionado1;
global M;
global L;
global indice_seleccionado2;
global ventana
indice_seleccionado2 = get(handles.listbox1,'Value');

if(handles.edit1<handles.edit2)
   Aa = -20*log10(handles.edit1);
else
   Aa = -20*log10(handles.edit2);
end
```

```
if(indice_seleccionado1==1)%Filtro paso alto
    %wc = handles.edit3 - (handles.edit3-handles.edit4)/2;
    valor = handles.edit3-handles.edit4;
    switch Aa
    case 1,
        Aa <= 21;
        %rect·ngular-->rectwin(L)
        M = ceil(4*pi/valor)-1;
        ventana = 1;
    case 2,
        Aa <= 44;
        %hanning-->hanning(L)
        M = ceil(8*pi/valor);
        ventana = 2;
    case 3,
        Aa <= 53;
        %hamming-->hamming(L)
        M = ceil(8*pi/valor);
        ventana = 3;
    case 4,
        Aa <= 74;
        %blackman-->blackman(L)
        M = ceil(8*pi/valor);
        ventana = 4;
    otherwise
        Aa>74;
        %kaiser-->kaiser(2*M+1,alfa)
        if(Aa<=21)
            beta = 0;
        elseif(Aa>=50)
            beta = 0.1102*(Aa-8.7);
        else
            beta = 0.5842*(Aa-21)^.4+0.07886*(Aa-21);
        end

        if(Aa<=21)
            M =ceil(5.794/valor);
        else
            M =ceil((Aa-7.95)/(28.72*valor));
        end
        ventana = 5;
    end

else
    %Filtro paso bajo
    wc = handles.edit3 + (handles.edit4+handles.edit3)/2;
```

```
    valor = handles.edit4-handles.edit3;

    switch Aa
    case 1,
        Aa <= 21;
        %rect·ngular-->rectwin(L)
        M = ceil(4*pi/valor)-1;
        ventana = 1;
    case 2,
        Aa <= 44;
        %hanning-->hanning(L)
        M = ceil(8*pi/valor);
        ventana = 2;
    case 3,
        Aa <= 53;
        %hamming-->hamming(L)
        M = ceil(8*pi/valor);
        ventana = 3;
    case 4,
        Aa <= 74;
        %blackman-->blackman(L)
        M = ceil(8*pi/valor);
        ventana = 4;
    otherwise
       Aa>74;
       %kaiser-->kaiser(2*M+1,alfa)
       if(Aa<=21)
          beta = 0;
       elseif(Aa>=50)
          beta = 0.1102*(Aa-8.7);
       else
          beta = 0.5842*(Aa-21)^.4+0.07886*(Aa-21);
       end

       if(Aa<=21)
          M =ceil(5.794/valor);
       else
          M =ceil((Aa-7.95)/(28.72*valor));
       end
       ventana = 5;
    end

end

L = M*2+1;
save valores L M indice_seleccionado2 ventana
```

```
 % --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%Error de aproximaciÛn en banda de paso
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
Val = get(hObject, 'String');
NewVal = str2double(Val);
handles.edit1 = NewVal;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%Error de aproximaciÛn en banda atenuada
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
Val1 = get(hObject, 'String');
NewVal1 = str2double(Val1);
handles.edit2 = NewVal1;
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%Frecuencia extrema de la banda de paso
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double
Val2 = get(hObject, 'String');
NewVal2 = str2double(Val2);
handles.edit3 = NewVal2;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
%Frecuencia extrema de la banda atenuada
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double
Val3 = get(hObject, 'String');
NewVal3 = str2double(Val3);
handles.edit4 = NewVal3;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double
Val4 = get(hObject, 'String');
NewVal4 = str2double(Val4);
handles.edit6 = NewVal4;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double
Val5 = get(hObject, 'String');
NewVal5 = str2double(Val5);
handles.edit7 = NewVal5;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

valor1 = handles.edit6;
valor2 = handles.edit7;


save variable valor1 valor2

botok=uicontrol('Style','pushbutton','Units','normalized','Position',[.84 .03 .12
.05],'String','CONTINUAR','Callback','clear all; close all;clc; gui2;');
```

3.2.4 Evaluación: Código de la segunda ventana de petición de resultados

```
function varargout = gui2(varargin)
% GUI2 MATLAB code for gui2.fig
%      GUI2, by itself, creates a new GUI2 or raises the existing
%      singleton*.
%
%      H = GUI2 returns the handle to a new GUI2 or the handle to
%      the existing singleton*.
%
%      GUI2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI2.M with the given input arguments.
%
%      GUI2('Property','Value',...) creates a new GUI2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before gui2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to gui2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui2

% Last Modified by GUIDE v2.5 20-Aug-2013 17:45:04

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui2_OpeningFcn, ...
                   'gui_OutputFcn',  @gui2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before gui2 is made visible.
function gui2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui2 (see VARARGIN)

% Choose default command line output for gui2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

background=imread('imagen2.jpg');%imagen de fondo
axes(handles.background);

axis off;

imshow(background);
handles.output=hObject;
guidata(hObject,handles);

% UIWAIT makes gui2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = gui2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1 contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from listbox1
```

```
global indice_seleccionado3;
indice_seleccionado3 = get(handles.listbox1,'Value');


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
open('practica2.pdf');


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ans=questdlg('¿Desea salir del programa?','SALIR','Si','No','No');
if strcmp(ans,'No')
return;
end
clear,clc,close all


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load variable;
load valores;
global indice_seleccionado3;
```

```
if (indice_seleccionado2==ventana)
    txt{1} = sprintf ('Respuesta 1: Correcta\n');
else
    txt{1} = sprintf ('Respuesta 1: Incorrecta\n');
end

if (valor1==M)
    txt{2} = sprintf ('Respuesta 2: Correcta\n');
else
    txt{2} = sprintf ('Respuesta 2: Incorrecta\n');
end

if (valor2==L)
    txt{3} = sprintf ('Respuesta 3: Correcta\n');
else
    txt{3} = sprintf ('Respuesta 3: Incorrecta\n');
end

if (indice_seleccionado3==4)
    txt{4} = sprintf ('Respuesta 4: Correcta\n');
else
    txt{4} = sprintf ('Respuesta 4: Incorrecta\n');
end

msgbox(txt, 'Resultados');
```

4. **Anexo: Enunciado de las prácticas**

Práctica 1: Análisis de funciones ventana

Objetivo: analizar las propiedades de las diferentes funciones ventana, tanto en el dominio del tiempo como en el dominio de la frecuencia y compararlas en función de los parámetros que las definen.

P1.1. Ventana rectangular (rectwin)

P1.2. Ventana triangular (triang, bartlett)

P 1.3. Ventana de Hanning (hanning, hann)

P1.4. Ventana de Hamming (hamming)

P1. 5. Ventana de Blackman (blackman)

P1. 6. Ventana de Kaiser (káiser)

P1.7. Comparativa

1. Representar las funciones ventana en el dominio del tiempo para las siguientes longitudes: L=23, 2L y 3L, en el intervalo de tiempo 0 a 100. En el caso de la ventana de Kaiser repetir empleando β 2 y 3. (Emplee la función "stem").
2. Representar en unidades lineales la transformada de Fourier, el módulo, de las funciones ventana del apartado anterior. (**El eje de frecuencias debe quedar correctamente definido en unidades de frecuencia**). (Emplee la función "plot")
3. Representar en unidades logarítmicas la transformada de Fourier, el módulo, de las funciones ventana del apartado anterior. (El eje de frecuencias debe quedar correctamente definido).
4. Indicar razonadamente cómo varía la anchura del lóbulo principal de la transformada de Fourier de las funciones ventana en función de su longitud. Si es posible obtenga una expresión. Compare con las expresiones tabuladas.
5. Indicar razonadamente cómo varía la atenuación de los lóbulos secundarios respecto al lóbulo principal de la transformada de Fourier de las funciones ventana en función de su longitud. En el caso de la ventana de Kaiser describa el efecto que tiene la variación del parámetro β. Si es posible obtenga una expresión. Compare con las expresiones tabuladas.
6. Represente un una misma gráfica, y para la misma longitud de 120 muestras, la transformada de Fourier de las funciones ventana indicadas anteriormente en unidades logarítmica.

7. Generar una señal sinusoidal de frecuencia Ω=0.1383 (rad) en el intervalo 0:1120. Generar una ventana rectangular y una ventana triangular de L=114 muestras en el intervalo 0: (L-1) y muestras nulas hasta 1120. Obtener dos señales resultado de multiplicar la señal sinusoidal por cada una de las funciones ventana (este producto es lo que se denomina enventanado). Representar el espectro de la señal resultante con la amplitud en unidades logarítmica y el eje horizontal en unidades de frecuencia (Emplee la función "fft" con 2048 muestras). Describir los resultados de emplear una y otra ventana.

ENUNCIADO DE LA PRÁCTICA 2.

Diseñe un filtro FIR paso alto mediante el método de la ventana de forma que cumpla con la siguiente plantilla de especificaciones:
Error de aproximación en banda de paso: δ_p =0.0313.
Error de aproximación en banda atenuada: δ_a =0.0010.
Frecuencia extrema de la banda de paso: Ω_p=1.8886 (rad).
Frecuencia extrema de la banda atenuada: Ω_a =1.6387 (rad).

Pasos a seguir:

1. Justifique las funciones ventana que puede emplear de entre las tabuladas.
2. Calcule la respuesta al impulso del filtro paso alto ideal. Indique el valor de la pulsación de corte elegida justificándolo.

Para cada una las funciones ventana que pueda emplear par diseñar dicho filtro:

3. Obtenga (con MATLAB) la respuesta en frecuencia del filtro diseñado. Represente su módulo (en unidades logarítmicas) y su fase. Dibuje líneas que delimiten la plantilla de especificaciones.
Se recomiendo el uso de la función "**freqz**". Para la representación de la respuesta de fase se recomienda emplear "**unwrap**".
Represente módulo y argumento en una misma gráfica.
4. Justifique si se trata o no de un filtro de fase lineal. En caso de serlo indique el retardo de grupo y relaciónelo con el orden del filtro.
5. Determine la anchura real de la banda de transición teniendo en cuenta la respuesta en amplitud.
6. Represente en una sola gráfica el módulo de las respuestas en frecuencia de los filtros que cumplen la plantilla de especificaciones. Dibuje las líneas que delimitan la plantilla de especificaciones.

Para el filtro que tenga mayor orden:

7. Disminuya el número de muestras de la respuesta al impulso del filtro diseñado aproximadamente a la mitad. Indique cuál ha sido la consecuencia.
8. Aumente el número de muestras de la respuesta al impulso del filtro diseñado al doble. Indique cuál ha sido la consecuencia.

Notas:

a) La respuesta al impulso ideal y la función ventana deben tener el mismo instante de tiempo de simetría para multiplicarlas muestra a muestra. Por ello se recomienda determinar primero la longitud de la función ventana y después obtener la respuesta al impulso del filtro ideal en el intervalo de tiempo apropiado.
b) Deberá subir un documento con los resultados obtenidos en la actividad establecida a tal efecto.
c) Indique como respuesta correcta "1" si ha comprendido el enunciado de la práctica, "2" en caso contrario.

5. **Guía de usuario**.

Con esta guía se prentende dar al alumno las primeras indicaciones para utilizar la herramienta de forma adecuada.

Para poder usar la herramienta correctamente el primero paso es resolver el enunciado de la práctica a través de Matlab, unas vez resuelta la práctica y con los resultados obtenidos al alcance de la mano es el momento de utilizar la herramienta.

Una vez abierta la herramienta, se procederá a la introducción de los resultados. Primero se introducirán los datos dados en el enunciado de la práctica, y a continuación un resultado en cada casilla, los cuales serán, en unos casos de tipo numérico y en otros casos se deberá elegir una de las opciones dadas en la propia ventana de la herramienta.

En esta imagen podemos ver un ejemplo de utilización:

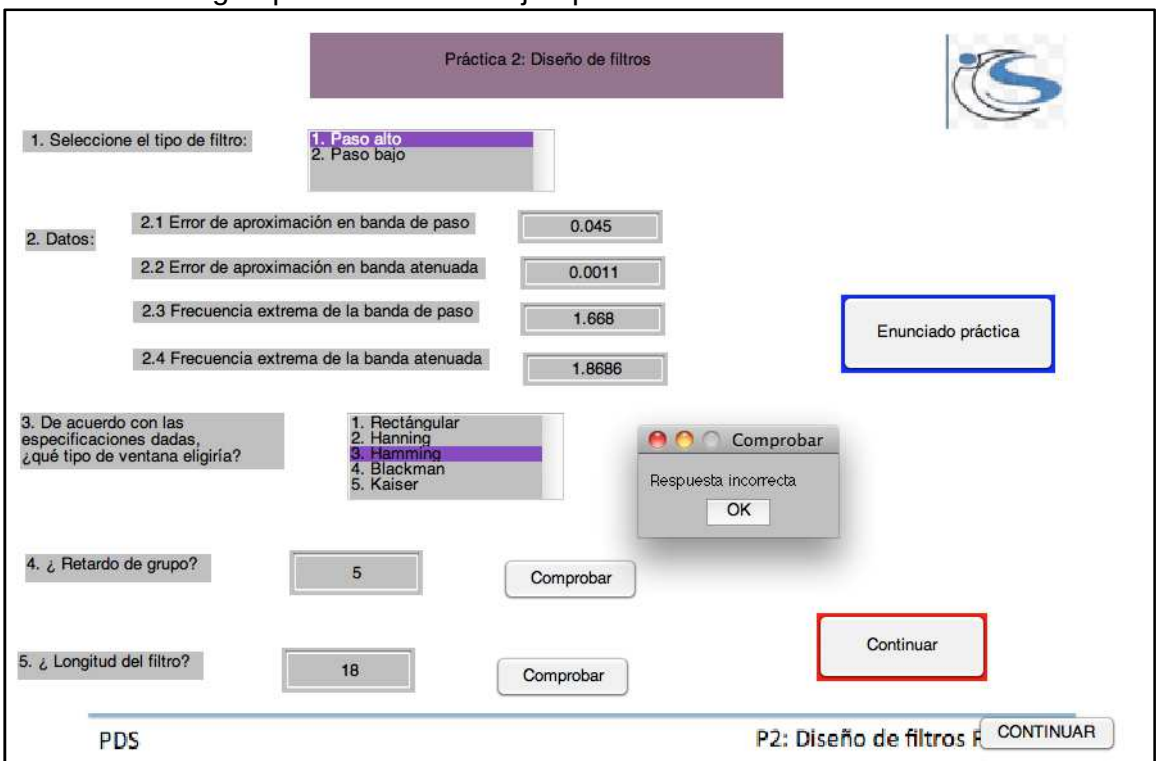


figura 24

5. Bibliografía

[1] *"Diferencias entre la formación basada en la red y la formación presencial tradicional".* Cabero, J.; López, E. (2009)

[2] "*Tecnología educativa. La formación del profesorado en la era de Internet*". Pons, Juan de Pablos (2009):

[3] Evaluación del método e-learning: Revista ELectronica de Investigación y Evaluación Educativa (Relieve) , *v. 9*, n. 2. (22 Oct. 2003)

[4] "*E-learning e internet como herramientas de autor para profesores de internet*". José Antonio Campos (2003). Biblioteca Centro Virtual Cervantes. (http://cvc.cervantes.es/ensenanza/biblioteca_ele/asele/pdf/14/14_0129.pdf)

[5] Moodle (http://es.wikipedia.org/wiki/Moodle)

[6] Libro de teoría Procesado Digital de la Señal. Autor: César Benavente Peces (EUITT)

[7] Aprende Matlab 6.0 como si estuviera en primero. Universidad de Navarra

[8] Matlab: Definiciones e historia (http://es.wikipedia.org/wiki/MATLAB)